Considerations on the Insularity of Performance Evaluation

Domenico Ferrari

Computer Science Division
Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California, Berkeley

ABSTRACT

The author argues that systems performance evaluation, in the first twenty years of its existence, has developed in substantial isolation with respect to such disciplines as computer architecture, system organization, operating systems, and software engineering. The possible causes for this phenomenon, which seems to be unique in the history of engineering, are explored. Its positive and negative effects on computer science and technology, as well as on performance evaluation itself, are discussed. In the author's opinion, the drawbacks of isolated development outweigh its advantages. Thus, the author proposes instructional and research initiatives to foster the rapid integration of the performance evaluation viewpoint into the main stream of computer science and engineering.

1. *Introduction*

The evaluation of computer systems performance enjoys a peculiar position in the science and engineering of computing. Its major peculiarity is that it exists as a discipline distinct from computer systems design, implementation, and management. To my knowledge, there are no other technical fields in which the evaluation of the performance of the machines those fields are concerned with is considered as a subject to be taught and researched in isolation. In the computer field, performance evaluation is the topic of courses offered by many universities all over the world, and has been recommended in model computer science curricula as a subject worth teaching [1],[2]; it is the area of specialization of a number of computer professionals and researchers, most of whom are members of such professional organizations as SIGMETRICS (the Special Interest Group on Measurement and Evaluation of the Association for Computing Machinery), CMG (the Computer Measurement Group), CPEUG (the Computer Performance Evaluation Users Group), and ECOMA (the European Computer Measurement Association); and it is the subject of a number of textbooks (see for example [3]-[13]) and of several journals (e.g., *Computer Performance, Performance Evaluation).* No

[1]The research reported herein was supported in part by a MICRO Grant from the University of California and the NCR Corporation. The ideas expressed in this paper are those of the author, and do not in any way reflect opinions or policies of the University of California or of the NCR Corporation.

similar activities and initiatives can be found as self-standing endeavors in such fields as electric transformer performance, combustion engine performance, digital circuit performance, industrial process performance, and so on. Even the case of operations research is different, since operations research does not have applications in industrial engineering only, and cannot be thought of as a discipline concerned only with the evaluation of industrial plant performance. In fact, some of its techniques can be used in computer systems performance evaluation and several other fields as well.

What are the reasons for this peculiar situation? Has it had beneficial or detrimental effects on the state of knowledge in the field, and on the development of computing technology and science? This paper will attempt to provide inevitably personal and subjective answers to the above questions. Section 2 briefly describes the isolated situation of the discipline, and some of the possible causes of such insularity. The advantages I see in the relatively isolated position of performance evaluation are summarized in Section 3, whereas Section 4 discusses those that are, in my opinion, the drawbacks of the situation. Recommendations for possible future action, based on the considerations in the preceding sections, are finally given in Section 5.

2. *The insularity of performance evaluation*

Understanding the meaning of the term *performance evaluation* requires that a definition of *system performance* be given, and that the term *evaluation* be interpreted as the assignment of *quantitative* values (to the extent that this is possible) to the indices of performance being considered. While the concept of performance is understood by most people as being strictly related to those of speed and efficiency, its precise definition can only be given by referring directly to the index, or indices, one chooses to express the performance of a system in quantitative terms. Thus, for example, in a certain study, the generic word *performance* may be used as a synonym of interactive response time; in another, as a short-hand name for a combination of response time and throughput rate; and so on.

The importance of performance and of its evaluation in all technical fields is obvious. Performance is one of the three fundamental categories of attributes that are indispensable for the viability of any technical system, the other two being *functionality* (with its equally important aspects of *correctness* and *reliability)* and *economicity.* A technical system, e.g., a machine, in order to be practically useful, must do what it is intended to do with acceptable continuity, reasonable efficiency, and affordable costs. Computer systems are no exception to these rules. Thus, the study of their performance aspects is an essential and fundamental component of computer engineering. I maintain that such study is vital also for computer science, since, when applied to existing systems, it must use the methods of the experimental sciences of nature. Indeed, the core of experimental computer science consists of some of the quantitative techniques, tools, and methodologies that are within the realm of performance evaluation.

Given these considerations, one would expect the elements of performance evaluation to be an integral part of any computer science curriculum, or, to be more precise, of any non-theoretical computer science course; to be present in the cultural background of any respectable computer scientist or engineer as one of the main dimensions of his or her professional universe; and to be paid adequate attention in most books and articles having to do with computer systems or software engineering. The situation we observe today, however, is quite different. The average level of "literacy" in performance evaluation among computer scientists, even the youngest generations of them, is quite low. The small community of professionals and researchers whose specialty is performance evaluation, rather than representing the tip of an iceberg, is in many respects the exclusive repository of a knowledge considered by most other computer scientists as highly specialized and of rather marginal importance. The subject is either taught in special courses, usually at the graduate level, or it is not taught; only in very few cases are serious, non-superficial performance considerations introduced when and where they ought to be, that is, in computer architecture, operating systems, computer design, distributed systems courses, rather than treated separately and out of their natural context. This situation would immediately look paradoxical if one tried to extend it to the other two fundamental categories mentioned above, namely, functionality and economicity. Imagine for instance an undergraduate course on computer architecture in which the subject of instruction sets would be ignored, only to be deferred to a graduate specialty course on CPU functionality which only a few older students could take; or a Special Interest Group on Cost Aspects in Computer Design (SIGCOST?); or a textbook on operating systems which would discuss the performance of various memory management policies without explaining what these policies are and how they work.

Why is the situation so different from what one would expect? Several plausible answers to this question may be proposed. First of all, the field of computers is still very young. In spite of its extremely fast, almost explosive, development, reaching scientific maturity requires, even in today's world, a non-negligible amount of time. Perhaps just because of the tumultuous progress that has characterized the field so far, there has been little incentive for reflection, and the quantitative evaluation of system performance certainly requires a more reflexive attitude than the introduction of new, more powerful functionalities. If computer systems are young, performance evaluation is much younger: its year of birth may be considered 1965, when Alan Scherr's classical Ph.D. dissertation [14] was submitted. Considerable progress has been made by the discipline in its first twenty years. Some of the milestones of this progress in the area of evaluation techniques have been, in measurement, the introduction of hardware monitors, of sampling software monitors, and of on-line system monitoring; in simulation, the advent of computer system simulation languages and packages, and the concept of regenerative simulation; in analytic modeling, the various solution techniques for separable queueing networks with multiple classes of processes and multiple chains, and the approximation methods for non-separable networks. In the fields of performance improvement *(tuning)* and configuration design, the identification of *bottlenecks* as the causes of the most common performance disease and the discovery of methods for diagnosing and curing it have been mainly responsible for the progress made since 1965. Installation management has benefited quite substantially from these advances in evaluation techniques and tuning. In particular, reliable capacity planning (i.e., the determination of the predicted time in the future when the capacity of an installation will become insufficient, and of the most cost effective way of upgrading that installation) has been made possible by progress in modeling and workload forecasting techniques. Finally, in software performance evaluation, an important advance has been due to the introduction of the concept of *program profile* and of techniques for the automatic profiling of programs.

Another possible reason for the isolation of performance evaluation is that computers are more complex than most other man-made machines, mainly because of the difficulty of quantifying the needs and the behaviors of their human users. Most other existing machines are operated by humans, but their performance can be much more easily characterized in ways independent (or almost) of human behavior: consider, as examples, an elevator, a crane, an electric motor, and a video cassette recorder. Even the performance of a multiple-user system like an electricity distribution network or a telephone network can usually be characterized in ways that are simpler than those required for a computer system's performance, probably because in the latter case the human users have a much larger number of options for their uses of the system. Progress toward a situation similar to those of older, better established branches of engineering is thus impeded by our ignorance about the needs and the behaviors of computer users in various types of environment.

Yet a third possible reason is that, perhaps as a consequence of the previous two points, a substantial fraction of computer scientists believe in the predominance of the "artistic" aspects of our field. While recognizing the fundamental importance (in ours as in all other technical disciplines) of human creativity, intuition, and imagination, I have serious difficulties convincing myself that computer systems are works of art (or black magic, as someone would propose), and therefore cannot and should not be subjected to quantitative evaluation. One might argue that this, rather than being a cause, is an effect of the underdevelopment of performance evaluation. The truth probably is that it is both a cause and an effect. Be that as it may, many people are always ready to welcome any argument they can use to justify their avoiding the toil of a scientific evaluation study, or of performance-driven design. And the more articulate among them can make these arguments almost credible.

The situation I have described, and of which I have tried to explain the main reasons, can be summarized by stating that, during the first twenty years of its life, performance evaluation has been characterized by its isolation from the "main stream" of computer science and engineering. The discipline has, in some sense, benefited from insularity, as we shall see in the next section. However, the drawbacks of the situation, to be discussed in Section 4, have been and are, in my opinion, even more evident.

3. *The benefits of insularity*

Even if performance evaluation had been one of the main concerns of hardware and software designers, implementors, managers, and users since the very beginning, the need for more powerful and more modern techniques, tools, and methodologies would have required a continuous concentration of research, development, production, and marketing efforts in its specific realm. However, one could make the point that its having been considered as a separate, specialized topic, with its peculiar problems and approaches, has caused these problems and approaches to be explored in greater depth; also, that its having often been divorced from a specific and pressing practical goal has made its results less dependent on technology and applications. In other words, the ease with which performance problems have been abstracted from their context, their "here and now", has helped emphasize the concepts and establish the foundations of the discipline. Insularity in the past might thus make integration in the future more effective and more advantageous.

There is, in my opinion, some truth in the above arguments, though the extent to which the achievements in performance evaluation have been positively influenced by its relative isolation is a matter of pure speculation. A direct consequence of insularity in the instructional sector has been the systematization the discipline has required in order to become teachable. Concepts, problems, techniques, tools and methodologies have been identified, defined, and classified. Structural properties have been detected, and the discovery of symmetries and analogies has led to interesting, sometimes useful, results.

In research, progress has been most impressive, especially during the last decade, in the area of analytic modeling, and particularly in that of queueing network models of computer systems and networks. If investigations had been motivated by specific architecture, operating systems, or installation management problems, knowledge in the queueing modeling area might now be less deep, or less extensive, or both. The question about whether such depth and extensiveness are useful in practice might be partially answered by stating that in some cases there is no reason to doubt the potential future usefulness of those queueing modeling results which do not seem particularly useful today. And, in any case, all new results have extended our knowledge. As we shall see in the next section, however, these conquests have not increased the popularity of the discipline. On the other hand, the appearance on the market, in the last several years, of performance analysis packages based on analytic (as well as simulation) approaches has tremendously enhanced the practical value of these techniques.

Other areas in which progress has been noticeable include measurement principles, techniques, and tools; tuning; and capacity planning. The insularity of performance evaluation has very often caused measurement to be an afterthought in system design, thereby favoring the emergence of instruments to be added from the outside to running systems. Since most computer manufacturers resisted for a long time the rather weak temptation of providing their customers with measurement tools, these instruments were produced by independent, specialized companies. The emphasis was (and is) on hardware monitors, because of their portability, or on software monitors for IBM or IBM-compatible systems, because of the popularity of such systems. The existence of a small but active and competitive "measurement industry" has certainly contributed to the development of the field, and stimulated the study of principles, the invention of new techniques, and the design of new tools. In the tuning and capacity planning areas, progress has been mainly based on the definition of bottlenecks and on the investigation of methods (both empirical and analytical) for detecting them. Substantial help, in these problems as well as in that of predicting the impact on performance of a tuning or upgrading action, has been provided by advances in modeling techniques.

4. *The drawbacks of insularity*

The development of performance evaluation as a relatively isolated discipline has also had, in my opinion, serious negative consequences. For convenience of discussion, the main drawbacks of autonomous development can be summarized by the following two statements:

(a) the separation between performance and functionality concerns has contributed to the establishment of what I would call a distorted mentality among computer scientists;

(b) the study of performance evaluation as an independent subject has sometimes caused researchers in the area to lose contact with reality.

That the mentality of a number of computer scientists is distorted becomes immediately evident as soon as one recognizes the importance of the performance evaluation viewpoint. As mentioned above, the emphasis on the "art" of computer design, of system management, of programming, and so on, is antithetic to the quantitative philosophy of performance evaluation. The performance evaluation viewpoint emphasizes the scientific method, well-planned and controlled experimentation, careful use of mathematical techniques for prediction; it does not favor serendipity as the ordinary approach to problem solving, "hacking", or the "hack now, fix later" methodology. The idea that the design of a system consists of implementing it, or of designing its functional aspects first, then implementing, and finally trying to improve the low-quality product of this procedure can be reluctantly accepted only as a dire necessity, dictated by ignorance, but cannot be glorified as the ultimate approach to system design. The absence of a widespread scientific mentality even among the researchers can be seen in the formulation of many research projects in computer science: only seldom are the hypotheses to be verified by a project clearly stated [15], the experiments to be performed carefully planned, and the criteria to be used to evaluate the project's results specified in advance. Note that hypotheses, plans, and criteria can and should be formulated even for projects in which quantification is difficult or impossible. Since most of the research proposals are subject to peer review, or handled by experts in the various disciplines, or both, we must conclude that the standards of the research community do not normally expect hypotheses, criteria, and experimental designs to be spelled out. In other words, the prevailing mentality differs quite substantially from that championed by the performance evaluation community.

Without denying the essential role in computer science (as well as in any other endeavor) of our non-rational faculties, I am tempted to observe that the "artistic" approach seems to be easier to use than the "scientific" one. It is also much faster, and speed is certainly crucial in a field where time is so precious. The members of the performance evaluation community are, unfortunately, at least partially responsible for the complexities of the scientific approach. Their relatively weak contacts with design, implementation, installation, and management realities have not stimulated them strongly enough to focus their efforts on making their methods truly usable and useful in practice. Something in this direction has been done, but is far from being sufficient. Because of the difficulties of the problems, the extremely fast pace of technological progress, the relatively small size of the performance community, and, alas, also the pursuit of rather abstract topics by some of the members of this community, the approaches and solutions offered by performance evaluation to the practitioners always tend to be late with respect to the developments in computing technology. For instance, reasonable models for the study of centralized systems and their workloads were barely becoming available when the world was already moving towards distributed systems, for which even a definition of performance has not been agreed upon yet, not to speak of the questions raised by their measurement and modeling, and by the characterization of their workloads.

Of course, this chronic time lag does not help promote the performance evaluation mentality, which, when not ignored altogether, is perceived as a cumbersome impediment rather than as a powerful viewpoint or a practically useful mental tool.

Another, even more serious problem caused by the lack of contact with reality that characterizes a number of self-styled computer systems performance modelers is the practical irrelevance of their efforts, which discredits the whole discipline in the eyes of designers, architects, systems programmers, and installation managers. A number of the models presented and analyzed in the literature do not provide any insight into the really important systems issues, which they do not even address. Such efforts sometimes advance the modeling methodology, when they introduce new solution techniques for queueing network models that are at least potentially useful. A substantial contribution to clarity would be made if those researchers whose primary interests are in mathematical techniques for solving queueing models ceased to be considered as members of the systems performance evaluation community, and were thought of as operations researchers or applied mathematicians. The criterion for distinguishing the two groups is extremely simple: the performance evaluator tries to solve computer systems problems, and uses the most appropriate techniques and tools at hand, which may include analytic models; the applied mathematician tries to advance knowledge in queueing theory and solution techniques, and uses computer systems problems to demonstrate the applicability of that theory and those techniques. It is no wonder that the problems selected as examples by persons who have little or no interest in computer systems often turn out to be off the mark. Unfortunately, even those results which would be applicable to real problems are sometimes not accessible to their potential users due to the forbidding mathematical shroud by which they are so effectively protected.

5. *The future*

There is little doubt in my mind that the drawbacks of insularity outweigh its advantages. However, undoing or modifying what has been done during the last twenty years is obviously impossible. Of greater interest is the question about whether something can and should be done in the future to strengthen the ties of the discipline with the rest of computer science, reduce if not eliminate the time lag, make its approaches and results truly useful and really used, and spread what I have described above as the performance evaluation mentality. I assume that the reader of this section believes these goals to be inherently good, or is interested anyway in my answers to the question. Thus, I will not try to prove that full integration of performance evaluation into the main stream of computer science is desirable, as I think this is largely a matter of taste. Personally, I would like to see a much greater integration than we have now, even to the point of dissolving the fundamental aspects of performance evaluation into such disciplines as computer architecture, operating systems, and software engineering. What should then be done to foster integration?

In my opinion. the most urgent actions to be undertaken are in the research sector. Much progress has been made, but much remains to be accomplished. The greatest obstacle to a quantitative treatment of computer systems is the absence of a universally accepted and carefully validated theory of such systems. This is perhaps the main reason for the low popularity of the performance evaluation viewpoint and for the isolation of the discipline. Other branches of engineering do not explicitly care about performance since they are based on theories, laws, equations which provide designers and users with quantitative tools for analysis and sometimes even for synthesis: for electrical, magnetic,. and electromagnetic systems, these are Maxwell's equations; for mechanical systems, the laws of dynamics and kinematics; for civil engineering structures, the equations of statics and the principle of virtual works; and so on.

The most interesting approaches that have been proposed for the establishment of a theory of computer systems are those based on queueing models and on operational analysis [16],[17]. The weakest aspect of these approaches is still their treatment of the workload characterization problem. Without clear definitions and sound methodologies, the workload of an installation cannot be quantitatively described, and, without such a description, performance evaluation is impossible or useless. It has been possible to translate elusive terms like "performance", "bottleneck", and "computing power" into precise, quantitative definitions: an effort should therefore be made to provide similar types of definitions for the "load" of a system, the "relative weight" of the various components of a workload, and the relationships between system-independent user-level requirements and system-dependent resource demands. We need to know what variables must be used to represent the workload of a particular installation; how the choice of these variables should be influenced by the system's organization, by the operating system, by the types of users, and by the applications mix; what variables can be used to characterize a workload in a system-independent way, and how can they be transformed into their system-dependent equivalents once knowledge about the specific system becomes available. System-independent characterizations of all major applications must be collected. Methods for dealing satisfactorily with the dynamics of workloads, the impacts of performance changes on user behavior, and the portability of measurement results from one environment to another need to be investigated. In my opinion, substantial progress in workload characterization is an absolute prerequisite to the establishment of a useful theory of computer systems and to all the other actions to be mentioned in the rest of this section.

Other research areas immediately require additional attention. To reduce the time lag between systems technology and performance evaluation, the extensions of the definitions of performance and workload to the worlds of distributed systems and supercomputers should be investigated, and practical models for such systems introduced as well as experimented with. On the more applied side, measurement tools and modeling packages should be coupled together, and integrated with one another.

A much greater involvement of performance evaluation experts into real-world projects is essential. Such projects should include designs and implementations of systems, configurations, operating systems, applications; management of installations; formulation of capacity plans; and tuning and upgrading studies. While leading or participating in these projects, the experts in performance evaluation should gather measurements and construct models which, to the extent that proprietary information can be disclosed, ought to be published. Relatively few case studies illustrating the practical applications of performance evaluation concepts, techniques, and tools have appeared in the literature. In publications, certain minimum scientific standards for papers and research proposals should be established and adhered to. No statements about the performance of a new system, policy, algorithm, or component should be made without appropriate empirical or modeling support. All the details about such support that are needed to reproduce the measurement or modeling experiments, and the rationale for the choices made in designing and running such experiments, ought to be given. Greater rigor and severity in applying the performance evaluation mentality on the part of the members of the performance evaluation community will go a long way towards spreading that mentality outside the community. However, substantial help must be given to those that are to be converted to the cause of performance evaluation: for instance, parametric workloads and workload models of known and realistic characteristics for the important applications should be defined, to replace the unrealistic benchmarks in current use, for example, in CPU speed comparisons.

In summary, I am advocating the advent of *applied performance evaluation* as the focus of most of the future research in the field. In the instructional sector, the main goal of the actions I am proposing is *integration*, which can also be seen as a form of applied performance evaluation. Those members of the community who belong to the academic world ought to wage war on the curriculum development front to obtain that performance evaluation topics, concepts, and techniques be introduced into undergraduate computer systems and software engineering courses. This result, however, will be a step in the right direction, but will not be sufficient: the integration one should strive for is fine-grained, capillar; no longer should syllabi merely include "performance evaluation topics", but the quantitative viewpoint should permeate the entire subject matter, which is now mostly treated in a qualitative, descriptive vein in many courses. A more effective approach would entail the actual teaching (perhaps once, or periodically, for demonstrative purposes) of systems courses by members of the performance evaluation community. Whether or not this will be possible depends on a host of local factors; however, an essential ingredient of such direct involvement, one which might even make it superfluous, is the preparation and successive distribution of exercises, to be employed in those courses, requiring the adoption of the performance evaluation viewpoint and involving the use of simple performance analysis techniques or tools (system instrumentation, program instrumentation. benchmarks, analytic modeling packages, and so on). These exercises could be used by the instructors as classroom, laboratory, or homework problems, and instructor's manuals containing their description and the necessary background material could be published. Perhaps an organization like SIGMETRICS could promote the collection and the widespread distribution of the exercises already being used in a number of universities in various systems courses, as well as stimulate the creation and the sharing of new ones.

Thus, even in teaching and the preparation of instructional aids. the transition from pure to applied performance evaluation will favor integration, and probably hasten the eventual disappearance of the esoteric cult of performance evaluation as well as of the small sect that has kept it alive but relatively secret for twenty years.

Acknowledgements

The author is grateful to Luis Felipe Cabrera, Ed Lazowska, Jehan-Francois Pâris, and Alan Smith for their comments on an early draft of this paper, and their precious suggestions.

He is also deeply indebted to Diana Barclay for her invaluable help in the preparation of the various versions of the manuscript.

References

[1] The *IEEE Computer Society Model Program in Computer Science and Engineering, IEEE Computer Society Press, Silver Spring, MD.* December 1983.

[2] R.H. Austing *et al.,* "Curriculum '78, Recommendations for the Undergraduate Program in Computer Science", *Comm. ACM,* vol. 22, n. 3, March 1979, pp. 417-166.

[3] M.E. Drummond, *Evaluation and Measurement Techniques for Digital Computer Systems.* Prentice-Hall, Englewood Cliffs, NJ, 1973.

[4] H. Hellerman and T.F. Conroy, *Computer System Performance.* McGraw-Hill, New York, NY, 1975.

[5] L. Svobodova, *Computer Performance Measurement and Evaluation Methods: Analysis and Applications.* American Elsevier, New York, NY, 1976.

[6] D. Ferrari, *Computer Systems Performance Evaluation,* Prentice-Hall, Englewood Cliffs, NJ, 1978.

[7] B. Beizer, *Micro-Analysis of Computer System Performance,* Van Nostrand Reinhold, New York, NY, 1978.

[8] H. Kobayashi, *Modeling and Analysis: An Introduction to System Performance Evaluation Methodology.* Addison-Wesley, Reading, MA, 1978.

[9] E. Gelenbe and I. Mitrani, *Analysis and Synthesis of Computer System Models.* Academic Press, New York, NY, 1980.

[10] C.H. Sauer and K.M. Chandy, *Computer Systems Performance Modeling.* Prentice-Hall, Englewood Cliffs, NJ, 1981.

[11] S.S. Lavenberg, Ed., *Computer Performance Modeling Handbook.* Academic Press, New York, NY, 1983.

[12] D. Ferrari, G. Serazzi, and A. Zeigner. *Measurement and Tuning of Computer Systems.* Prentice-Hall, Englewood Cliffs, NJ, 1983.

[13] E.D. Lazowska, J. Zahorjan, G.S. Graham, and K.C. Sevcik, *Quantitative System Performance.* Prentice-Hall, Englewood Cliffs, NJ, 1984.

[14] A. L. Scherr, "An analysis of time shared computer systems", Ph.D. Dissertation, MIT, 1965; also MIT Press, Cambridge, MA, 1967.

[15] P.J. Denning, "What is experimental computer science?", *Comm. ACM,* vol. 23, n. 10, October 1980, pp. 543-544.

[16] J.P. Buzen, "Fundamental operational laws of computer system performance", *Acta Informatica,* vol. 7, n. 2, 1976, pp. 167-182.

[17] P.J. Denning and J.P. Buzen, "The operational analysis of queuing network models", *ACM Computing Surveys,* vol. 10, n. 3, September 1978, pp. 225-261.