Ubiquity Symposium

The Science In Computer Science

Performance Analysis: Experimental Computer Science at Its Best

by Peter J. Denning

Editor's Introduction

In the third installment of this symposium, which originally appeared in the *Communication the ACM*, we go back to 1981. More than 30 years ago, I called on ACM members to employ more experimental methods and avoid confusing hacking (tinkering) with true science. Joining a long tradition of ACM Presidents speaking up about computing as science.

Peter J. Denning
Editor

Ubiquity Symposium

The Science In Computer Science

Performance Analysis: Experimental Computer Science at Its Best

by Peter J. Denning

What is experimental computer science?[1] This question has been widely discussed ever since the Feldman Report was published in 1979 [1]. Surprisingly, computer scientists disagree on the answer. Some believe that it is large system development projects, i.e., computer and software engineering. Some believe that it is all the nontheoretical activities of computer science, especially those conferring "hands-on" experience. Some believe that computer science is still too young and no operational definition is yet available.

I disagree. There are well-established standards for experimental science. The field of performance analysis meets these standards and provides the best examples of experimental computer science.

Hypotheses, Apparatus, and Tests

Science classifies knowledge. Experimental science classifies knowledge derived from observations. The experimenters set up an apparatus, use it to collect data about a phenomenon, and then analyze the data to sustain or refute hypotheses about that phenomenon. The result of one line of investigation may be a model that becomes the apparatus for a later line of investigation.

[1] Part of this essay is based on my editorial, "What is Experimental Computer Science?" *in Communications of the ACM,* October 1980, pp. 543-544. The rest is based on my speech at the ACM SIGMETRICS Symposium on Performance Modeling, September 15, 1981.

The experimental apparatus may be a real system, a subsystem, or a model. The hypothesis may concern a law of nature, characteristics of people, design principles of computers, or the quality of models.

The key affects of experimental science are an apparatus for collecting data, a hypothesis, and systematic analysis to see whether the data supports the hypothesis. There is considerable latitude in the types of hypotheses and apparatuses that may be used.

Two Examples

It is no accident that the best examples of experimental computer science can be found in the field called performance analysis. The primary aim of this field is the construction, validation, and evaluation of computer-system models that are robust enough to be used for prediction. I will cite two examples of experimental science in this field. The first is the M44/44X project at the IBM T.J. Watson Research Lab in the middle 1960s; this project evaluated concepts of time sharing, especially memory policies and program behavior, by implementing and measuring them. This example illustrates experimental work in computer systems architecture. The second example is the study of queueing network models since 1971; this line of investigation illustrates how strong interaction between theory and experiment can lead to a conceptually simple model that may serve as the starting point for future lines of investigation. This example illustrates how yesterday's theorems can become tomorrow's definitions.

The M44/44X Project

The M44/44X project was conducted at the IBM Research Center in Yorktown Heights, N.Y., in the middle 1960s. Its purpose was to evaluate the emerging concepts of time-sharing systems by reducing them to practice and measuring them. The central principle of its architecture was a set of virtual machines, one for each user. The main machine was an IBM 7044 (M44 for short) and each virtual machine was an experimental image of the 7044 (44X for short). Virtual memory and multiprogramming were used to implement the address spaces of the 44Xs in the memory hierarchy of the M44. This machine served as the apparatus for numerous experiments about memory policy and program behavior.

O'Neill (1967 [2]) described the system architecture and early experience with it. It is interesting that they recognized the problem of thrashing and solved it with a load controller.

Les Belady conducted a series of projects to understand the behavior of paging algorithms, the effects of page size, and the costs of storage fragmentation. His comprehensive paper (1966

[3]) significantly increased our knowledge of paging algorithms applied to individual programs in fixed allocations of main memory. Belady studied haft a dozen policies and their variants. He concluded that the LRU (least recently used) policy is better than the FIFO (first in first out) policy, but that a simple variant bf FIFO based on usage bits gave a good approximation to LRU. He invented the optimal algorithm (MIN) and compared it with the realizable ones. He measured the effects of page size on performance, including the amount of superfluous information on pages (words not referenced after pages are loaded in main memory). His study was a model for similar experiments in other systems that independently corroborated his basic findings.

With multiprogramming, Belady discovered that system performance is improved by varying the space allocated to individual programs: Variable partitioning is more efficient than fixed. He and Carl Kuehner (1969 [4]) proposed a model for this that exploited that concave-up shape of the lifetime curve of a program. (The lifetime curve gives the mean virtual time between page faults when a given amount of space is allocated to the program.)

In joint work with Robert Nelson and Jerry Shedler (1969 [5]), Belady observed that the FIFO policy has anomalous behavior—that is, it may increase paging in response to increased space allocation. (They demonstrated that adding one page to memory may double the paging rate.) This work influenced the later work of Mattson *et al.* (1970 [6]), whose "stack algorithms" are well behaved.

In studying storage utilization, Brian Randell concluded that internal fragmentation is more serious than external fragmentation (1969 [7]). (Internal fragmentation is storage loss due to rounding up segments to integral numbers of pages; external fragmentation is storage loss due to holes in memory being smaller than segments awaiting loading.) Randell proposed a new addressing mechanism, called partitioned segmentation that allocated a large segment as a "head" consisting of pages and a "tail" consisting of a segment shorter than the page size.

Meanwhile, David Sayre and his colleagues studied the *virtual memory hypothesis:* The performance of properly tuned automatic memory management is better than the performance of the best handcrafted manual overlay strategy (1969 [8]). They compared programs run on the M44 with the automatic memory manager on and off to conclude that the hypothesis is correct for programs exhibiting locality of reference. This set of experiments laid to rest the remaining doubts about the efficacy of virtual memory.

Since the time of the M44 experiments, approximately 200 researchers around the world have contributed to the experimental effort to understand and optimize virtual memory operating systems (Denning 1980 [9]). Aside possibly from the experimental work by Yon Bard and his

colleagues on the CP-67 and VM/370 operating systems at the IBM Scientific Center in Cambridge, Massachusetts, I am not aware of any similar project in the public domain. The M44/44X project is a unique milestone along the highway of experimental computer science.

Queueing Network Models

The theory of stochastic queueing networks was developed in the 1950s and 1960s by Jackson, Gordon, and Newell [10]. This theory captured the interest of the computing community in 1971, when Jeffrey Buzen pointed out their application to the central server system and showed how to efficiently compute standard performance metrics in this model [11]. Since that time, a significant portion of the systems modeling community has been studying the *queueing network hypothesis:* The queueing network model is an accurate, robust description of standard performance metrics for a wide class of common computer systems.

The principal result of the Jackson-Gordon-Newell theory is that the steady-state probability of seeing a given apportionment of jobs among the devices of the system is of the *product form—it* is a series of factors, each depending on the number of jobs present at a given device and on the parameters of that device. Buzen discovered a simple recursion formula for calculating performance metrics, such as throughput and response time, at each device (1973 [12]).

Because the algorithms for these formulae are compact and efficient, they are easy to program, even on hand calculators. Validation studies soon revealed that the model will typically estimate the actual utilization to within 5 percent and the mean queue length to within 25 percent [10]. The accuracy of these models is now so well trusted that most analysts suspect an error in the model or its parameters if the formulae do not produce answers within these tolerances.

In 1975 Baskett, Chandy, Muntz, and Palacios extended the product form solution to include multi-class networks [13]. The computational algorithms were extended for this case by Reiser and Kobayashi (1975 [14]).

By this time the models were being applied to large problems—many customers, stations, and job classes. Numerical instabilities were encountered in the algorithms for these cases. In 1978 Reiser and Lavenberg [15] proposed *mean value analysis,* new recursions that avoided these problems. In using these equations, many intermediate values will be calculated and discarded en route to the mean values of response time and queue length for a large load. So Yon Bard, in consultation with Paul Schweitzer, proposed an approximation for the Reiser-Lavenberg

equations that gave excellent results [16, 17]. These equations are shown in the box at the right. [See Figure 1]

The first equation has a simple intuitive explanation (the response time is the mean service time per customer multiplied by the mean number of customers just after an arrival); the second equation is a law. A growing body of experimental evidence shows that these equations are sufficiently accurate and robust for many common systems [18, 19]. More experimental work is needed to precisely identify the class of systems for which these equations are good approximations.

I have now arrived at my main point: Because they are intuitive and validated, the Bard-Schweitzer equations can be used as the starting point for a theory of queueing networks. Much of our understanding, experimenting, and experience of fifteen years with queueing networks has been distilled into a fine essence, captured by these equations. Novices can use them wisely and well. This could not have happened without constant, complementary interaction between theoreticians and experimenters. We are witnessing a great scientific achievement.

Conclusion

The established standards of science can be used to distinguish true experimental research from engineering development projects in our field. The specialty of performance analysis contains excellent examples of experimental science. I have dwelt here on two. The M44/44X project is a paradigm of experimental study to evaluate the architecture of computer systems. The evolution of queueing network models is a paradigm of the interaction between theory and experiment, demonstrating that yesterday's theorems are tomorrow's definitions.

Although I have said that the systems modeling specialty contains the best examples of experimental computer science, I do not mean to imply that all work in this specialty is exemplary or that no work in other specialties is in the true spirit of experimental science.

In emphasizing that development projects are not necessarily experimental science, I do not mean to downgrade such projects. These projects can make substantive contributions to computer and software engineering, which are as important to the computing field as theory and experiment.

References

[1] Feldman, J., Editor. Rejuvenating experimental computer science. *Comm. ACM* 22, 9 (Sept. 1979), 497-502. See also the ACM Executive Committee position, same issue, pages 503-504.

[2] O'Neill, R.W. Experience using a time sharing multiprogramming system with dynamic address relocation hardware. *Proc. AFIPS Computer Conference* 30 (1967 SJCC), pp. 611-621.

[3] Belady, L.A. A study of replacement algorithms for virtual storage computers. *IBM Systems J.* 5, 2 (1966), 78-101.

[4] Belady, U.A. and Kuehner, C.J. Dynamic space sharing in computer systems. *Comm. ACM* 12, 5 (May 1969), 282-288.

[5] Belady, L.A., Nelson, R.A., and Shedler, G.S. An anomaly in the space-time characteristics of certain programs running in paging machines, *Comm. ACM* 12, 6 (June 1969), 349-353.

[6] Mattson, R.L., Gecsei, J., Slutz, D.R., and Traiger, I.W. Evaluation techniques for storage hierarchies, *IBM Systems J*. 9, 2 (1970), 78-117.

[7]. Randell, B. A note on storage fragmentation and program segmentation. *Comm. ACM* 12, 7 (July 1969), 365-369.

[8] Sayre, D. Is automatic folding of programs efficient enough to displace manual? *Comm. ACM* 12*,* 12 (Dec. 1969), 656-660.

[9] Denning, P.J. Working sets past and present. *IEEE Trans. Software Engineering* SE-6, 1 (Jan. 1980), 64-84.

[10] Denning, P.J., and Buzen, J.P. The operational analysis of queueing network models. *Computing Surveys* 10*,* 3 (Sept. 1978), 225- 261.

[11] Buzen, J.P. Analysis of system bottlenecks using a queueing network model. *Proc. ACM SIGOPS Workshop on Systems Performance Evaluation,* ACM, 1133 Avenue of Americas, NY 10036 (1971), pp. 82-103.

[12] Buzen, J.P. Computational algorithms for closed queueing networks with exponential servers. *Comm. ACM* 16*,* 9 (Sept. 1973), 527- 531.

[13] Baskett, F., Chandy, K.M., Muntz, R.R., and Palacios, F.G. Open, closed, and mixed networks with different classes of customers. *J. ACM* 22*,* 2 (April 1975), 248-260.

[14] Reiser, M., and Kobayashi, H. Queueing networks with multiple closed chains: theory and computational algorithms. *IBM J.R. &D.* 19 (May 1975), 283-294.

[15] Reiser, M., and Lavenberg, S.S. Mean value analysis of dosed multichain queueing networks. *J. ACM* 27, 2 (April 1980), 313-322.

[16] Bard, Y. Some extensions to multiclass queueing network analysis. *Proc. 4th Int'l. Symposium on Computer Performance Modeling, Measurement, and Evaluation,* H. Beilner and E. Gelenbe, Eds., North-Holland Publishing Co., Amsterdam, The Netherlands (1979).

[17] Buzen, J.P., and Denning, P.J. Measuring and calculating queue length distributions. *IEEE Computer* 13, 4 (April 1980), 33-44.

[18] Chandy, K.M., and Sauer, C.H. Computational algorithms for product form queueing networks. *Comm. ACM* 23, 10 (Oct. 1980), 573-583.

[19] Chandy, K.M., and Neuse, D. Linearizer: A heuristic algorithm for queueing network models of computing systems. *Comm. ACM,* scheduled to appear February 1982.

Figure 1. Bard-Schweitzer Equations

$R_k = S_k \left(1 + \frac{N-1}{N} Q_k\right), \qquad k = 1, \ldots, K$

$Q_k = N V_k R_k / \sum_{i=1}^{K} V_i R_i, \qquad k = 1, \ldots, K$

where

R_k = Mean response time per visit to device k

Q_k = Mean queue length at device k

S_k = Mean service time per visit to device k

V_k = Mean number of visits per job to device k.

N = Total number of jobs in system

K = Total number of devices in system

$Q_k (N - 1) / N$ = Queue length seen by arrival to device k

Equations iterated until Q_k converges; initial guess is $Q_k = N/K$.

About the Author

Peter J. Denning (pjd@nps.edu) is Distinguished Professor of Computer Science and Director of the Cebrowski Institute for information innovation at the Naval Postgraduate School in Monterey, California, is Editor of ACM *Ubiquity*, and is a past president of ACM.

DOI: 10.1145/2434536.2406361

Reprinted with permission from **Communications of the ACM*, Vol. 24 No. 11***
DOI: 10.1145/358790.358791

