Title
AN ABSOLUTE STANDARD FOR MEASURING COMPUTER SYSTEM PERFORMANCE

Author
Stevens, David F.

Publication Date
1974-05-22

Presented at the Computer Performance
Evaluation Users' Group Meeting,
Atlanta, Georgia, May 21-23, 1974.

LBL-1569 Rev.

AN ABSOLUTE STANDARD FOR MEASURING COMPUTER SYSTEM PERFORMANCE

David F. Stevens

May 22, 1974

Prepared for the U. S. Atomic Energy Commission
under Contract W-7405-ENG-48

LBL-1569 Revised

AN ABSOLUTE STANDARD FOR MEASURING COMPUTER SYSTEM PERFORMANCE*

David F. Stevens

Lawrence Berkeley Laboratory
Berkeley, California 94720

May 22, 1974

ABSTRACT

The measurement and evaluation of computer systems has recently become the second most popular indoor sport within the the computing community, but there has been essentially no attempt to establish a useful absolute standard against which the performance of a general purpose computer system can be measured. This paper discusses the desirability of such a yardstick and proposes a specific standard; its strengths and weaknesses are considered, and some suggestions for future refinement are presented.

Key Words and Phrases: system measurement, system evaluation, performance measurement, performance evaluation, computer system performance, measurement, evaluation.

CR Categories: 2.40, 2.44, 4.30, 6.20

*Work performed under the auspices of the U. S. Atomic Energy Commission

INTRODUCTION

The practice of computer system performance measurement and evaluation continues to become more prevalent, fostered not only by local concerns for the elimination of production bottlenecks, but also by federal encouragement. As more and more installations come to recognize the very real benefits of a continuing measurement and evaluation program, and as federal encouragement ripens into federal insistence, the trend will continue to accelerate. The result will be a plethora of reports and case studies of extreme particular significance, but admitting of no general comparisons. It would seem to be desirable to develop one (or possibly several) yardsticks against which arbitrary systems could be measured--yardsticks which are relatively independent of manfacturer, (hardware or software) architecture, system size, and CPU speed. Without such absolute measures different systems will remain essentially non-comparable. One possible yardstick, a standard unit for measuring system performance, is proposed below; but before considering the unit itself I would like to emphasize three attributes which such a standard should possess.

To begin with, a standard unit should be intuitively acceptable: the computing community is an independent lot and would resist the imposition of an "unreasonable" standard. This means it must be related to throughput, for throughput is recognized as the key to performance. Secondly, a standard unit should be simple: the measurement of an arbitrary system against the standard should be a straightforward operation, and the results should be difficult to misinterpret (or to misrepresent). And thirdly, a standard unit must provide a measurement which is in some sense independent of the size and power of the system being measured, so that meaningful comparisons of different systems can be achieved. (In this respect, the unit should be somewhat like gas mileage for automobiles: providing a useful comparison of performance, but depending upon the efficiency of the engine--and the driver!-- rather than upon the maximum speed attainable.)

MEASURING THROUGHPUT

We begin by considering throughput. "Throughput" is a rather loose term for the amount of work done by a computing system; it is a function of the number of simultaneous user processes and the speed at which each is carried out. Since we wish our final yardstick to be independent of speed, we begin with the problem of counting the number of user processes.

Before we can count them, however, we need some criteria to tell us which processes to count, for much of what goes on in a modern computing system is done, not for the end user, but for the computer center management (accounting, statistics gathering), or for the operations staff (console support, label verification), or to overcome deficiencies in hardware or software design. I propose the following definition:

> A "user process" is an I/O operation or string of computations which satisfies the following two conditions:
> (1) It advances the progress of a user job; and
> (2) It was explicitly requested by the user.

Thus, for instance, a rollout/rollin cycle is not a "user process" (even if requested by the user: it does not advance the progress of his job); nor is page swapping; nor system activity to process I/O interrupts (even though the I/O itself may be a user process). Spooling and staging operations to handle unit record (or other relatively slow) equipment are not user processes; that distinction is reserved for user access to the staged files. (All of these non-examples of user processes are system processes, designed to simplify the user's life or to improve the efficiency of the system; in other words, not all beneficent processes are user processes.) Furthermore, although it is usually necessary to occupy core in order to activate a user process, mere core occupancy itself is not a user process.

(Note: This definition was derived with batch processing in mind, but it applies as well to the interactive situation; however, one must avoid the temptation to equate the number of simultaneous user processes with the number of terminals currently on-line.)

Assuming now that we can count user processes, we must consider how to weight them. The simplest would be to give each process the same weight, but that seems unjust; a job outputting directly to an on-line printer progresses more slowly than a job spooling output to disk, even though the latter involves more overhead. Therefore it seems reasonable to assign weights to I/O channels which are proportional to the observed user-data transfer rates across the channels. Similarly, if there are multiple CPUs of differing speeds, they should be weighted accordingly. The difficulty comes in comparing CPU to channel.

Before attempting that, however, it might be instructive to consider what we expect of an ideal system. An ideal system is perfectly balanced, so that the CPU and all channels are fully utilized, all of memory is occupied, and every job in memory has something going for it--either CPU or a channel. Thus the total I/O time equals the total compute time. If we knew how much I/O to expect for each compute-second, then, we should be able to derive a channel-weighting factor.

Many years ago, in the process of a channel-analyzer survey of 7090 and 7094 installations, IBM did just that, determining that one bit of I/O occurred for each CPU instruction executed. (This ratio, known as Amdahl's Constant, seemed to hold true regardless of the nature of the installation or of the kind of work being done on the computer.) In the absence of any more recent data we use Amdahl's Constant to determine each channel's "Amdahl Factor": its observed user-data transfer rate (in bits/second) divided by the CPU speed (in instructions/second). Thus, for example, a 9-track, 1600 bpi tape moving at 200 ips has a data transfer rate of 2.56 megabits/second, and hence should have an Amdahl throughput weighting factor of U*2.56/M, where M is the CPU speed in MIPS, and U is user-utilization of the channel.

THE STANDARD UNIT

The weighted sum of the user-utilization of CPU and channels satisfies most of our requirements for a standard unit: it is simple and independent of speed, yet is clearly related to throughput; it lacks only independence of size. This is achieved by dividing by the size in megabytes to obtain our unit: simultaneous-(user)-processes-per-megabyte, or SPPM.

Now that we have it, what do we do with it? We use it as a lever to pry better performance out of existing systems and better systems out of reluctant manufacturers: we run contests (as we used to do with CPU utilization figures); we compare systems on the basis of SPPM; we establish acceptability thresholds. We use it to force the recognition that the utilization of <u>all</u> channels (not just the CPU) is important, and that it is important to know how much of that channel utilization is for the benefit of the User, and how much is for the benefit of the System.

PITFALLS AND LIMITATIONS

SPPM is an attempt to provide a general system measure useful in the multithread environment; it is limited and perhaps a little treacherous. Its simplicity may be somewhat chimerical in that it is easier to define than to determine. (Many installations would be hard put to determine the user-utilization (as opposed to the utilization) of most channels.) Amdahl's constant may no longer be valid, but until someone finds its current equivalent, it seems to be the best place to start. No allowance is made for heirachical memories, whether or not the lower levels are executable. And finally, SPPM suffers from a defect common to all utilization measures: a tendency to report inefficient usage as high utilization.

None of these faults is either minor or trivial, but the only way to develop an adequate standard is to begin with a possibly inadequate one and refine it; SPPM is a reasonable beginning.

TECHNICAL INFORMATION DIVISION
LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720