**Title**
OBSOLETE AN ABSOLUTE STANDARD FOR MEASURING COMPUTER SYSTEM PERFORMANCE




**Author**
Stevens, David F.



**Publication Date**
1973

Submitted to National Computer
Conference, New York, NY,
June 4-8, 1973



# AN ABSOLUTE STANDARD FOR MEASURING COMPUTER SYSTEM PERFORMANCE

David F. Stevens



January 1973

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

# DISCLAIMER

 LBL-1569

# AN ABSOLUTE STANDARD FOR MEASURING COMPUTER SYSTEM PERFORMANCE*

by David F. Stevens
Lawrence Berkeley Laboratory
Berkeley, California

AEC Contract No. W-7405-eng-48

January 17, 1973

*Work performed under the auspices of the United States Atomic Energy Commission

# AN ABSOLUTE STANDARD FOR MEASURING COMPUTER SYSTEM PERFORMANCE

David F. Stevens

Lawrence Berkeley Laboratory
Berkeley, California 94720

January 17, 1973

## ABSTRACT

The measurement and evaluation of computer systems has recently become the second most popular indoor sport within 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.

Kew 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

## INTRODUCTION

The current trend in the measurement of computing systems seems to be directed towards increased specificity -- instead of determining that a channel is busy, for instance, one determines that Channel 3 is busy reading a tape. This reflects the current principal motivation for measurement: improvement in performance achieved by detecting and eliminating system bottlenecks. While this is certainly a useful pursuit, the mass of detail collected is difficult to combine into a meaningful measurement of the total system in a form suitable for comparison with other systems, perhaps with radically different hardware or operating system philosophies. This situation will persist until the computing community as a whole adopts a standard unit against which arbitrary systems can be measured. Before proposing my candidate for the position of standard unit, I would like to emphasize three attributes which such a standard should possess.

First, the standard unit must be intuitively acceptable (to insure adoption by the whole of the computing community; if it is not adopted by the whole of the community, it is not a standard); hence, it must measure throughput, for throughput is the key to performance. Second, the standard unit must 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 third, the 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, but depending upon the efficiency of the engine -- and the driver -- rather than upon the maximum speed attainable.) Thus, the basis of our standard unit is a simple measure of throughput; if the first cut is not independent of size and power, it can always be normalized.

## A MEASURE FOR THROUGHPUT

Throughput is a function of the number of simultaneous user processes and the speed at which each process is carried out. Measuring the speed of the various processes is a well-understood art and within the capability of many of the instrumentation systems now available. To provide a measure for throughput, then, we must decide which processes to measure (i.e., determine which processes are user processes), and then combine the measurements for dissimilar processes into a meaningful mélange.

The decision is an arbitrary one, accomplished in accordance with some reasonable definition; the following may be adequate:

> 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 roll-out/roll-in cycle (even if requested by the user) is not a "user process"; nor is page swapping, nor system activity to process I/O interrupts (even if the I/O itself is a user process). Also, although the user may call for printing and card reading, spooling and staging operations are not user processes; that distinction is reserved for user accesses to the staged files. (These non-examples of user processes are system processes, designed either to simplify the user's life or to improve the efficiency of the system.) Furthermore, although it is in general necessary to occupy core to activate a user process, mere core occupancy itself is not a user process.

(Note: The preceding definition was derived with batch (or remote batch) operations in mind, but it applies equally well to the interactive situation; it is just that one must avoid the temptation to equate the number of simultaneous processes with the number of terminals currently on-line.)

The problem of neatly blending the dissimilar measurements is solved by weighting the various processes, assigning 1 to (full utilization of) the fastest CPU and appropriate factors to other CPU's and the channels, the latter depending upon observed transfer rates. Just how these factors should be determined is not yet altogether clear, but Amdahl's Constant (one bit of I/O per CPU instruction executed) provides a reasonable starting point -- i.e., for an M-mips CPU, the channel weights are $8*Bi/M$, where the Bi are the observed transfer rates in megabytes/second.

## THE STANDARD UNIT

The throughput measure described above already satisfies most of our requirements for the standard unit: it measures throughput, it is simple (we will return to this in a moment), and it is independent of speed. It requires only normalization (division by memory size in megabytes) to produce the standard: simultaneous-(user)-processes-per megabyte, or SPPM. Now 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.

The most obvious ways are probably the best ways: we run contests, as we did in the old days with CPU utilization figures and are now doing with increasingly esoteric monitoring data; we compare systems; we establish acceptability threshholds. It is still the case, for instance, that core is the most expensive component of many (most? ) computing systems; adoption of SPPM as a standard measurement unit will do more to promote efficient systems use of core than any number of resolutions passed at user-group meetings. Adoption of SPPM will force the recognition that the utilization of all channels (not just the CPU) is important, and that it is important to know how many of those words tossed back-and-forth are advancing user processes. Adoption of SPPM may even slow down what passes for progress: a well-run 7090 installation would have achieved an SPPM level of about 8 -- how many of us are even close to that now? (of course, we run bigger problems, but are they that much bigger? ? )

## PITFALLS AND LIMITATIONS

SPPM is a first attempt to provide a general system measure useful in the multithread environment; it is limited and perhaps a a little treacherous. Its simplicity may be somewhat chimerical in that it is much easier to state than to accomplish. Thus, although most installations could estimate the effective user bit-rate across a given channel, few could measure it with accuracy. Amdahl's Constant may not be the right starting point for channel weightings: it was determined in the early Channel Analyzer days, and is probably too generous for today's hardware but until someone refines it, it's the best thing available. No allowance is made for heirarchical 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 an inadequate one and refine it; SPPM is a suitable beginning.