Title
HOW TO INTERPRET COMPUTER PERFORMANCE MEASURES: A GUIDE FOR UPPER MANAGEMENT

Author
Stevens, David F.

Publication Date
1977-03-01

Submitted to Harvard Business Review

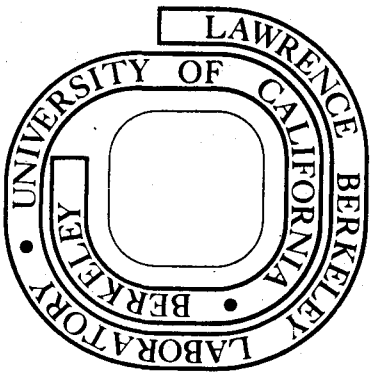
LBL-6115
Preprint c.1

HOW TO INTERPRET COMPUTER PERFORMANCE MEASURES: A GUIDE FOR UPPER MANAGEMENT

David F. Stevens

March 1, 1977

Prepared for the U. S. Energy Research and Development Administration under Contract W-7405-ENG-48



DISCLAIMER

LBL-6115

How to Interpret Computer Performance Measures: A guide for upper management*

David F. Stevens

Lawrence Berkeley Laboratory
University of California
Berkeley, California

March 1, 1977

Introduction

During the early stages of a company's involvement with computers, the EDP function is often relatively independent. As it increases in scope and influence, however, it tends more and more to become an object of upper management interest and control.[1] A necessary concomitant to effective control is a realistic system of measuring current performance. Unfortunately, upper management frequently lacks the technical background to develop suitable measures of its own, and must rely upon the EDP manager for suggestions. The EDP manager, in turn, often does not have a set of suitable measures ready to hand, and so he either adopts and adapts existing measures or seeks the advice of the manufacturer of his major computing system.

At this point it should be noted that neither the EDP manager nor the computer manufacturer is a disinterested party with respect to computer performance measurement: both are strongly motivated to demonstrate

---

* This work was done with support from the United States Energy Research and Development Administration.

[1] Gibson and Nolan, Managing the four states of EDP growth, HBR, January-February 1974.

effective and efficient use of the computing system. (From the EDP manager's point of view, computer system performance is an outward and visible sign of his own job performance; from the manufacturer's point of view, good experience [performance] with current equipment helps to open the door to further sales.) Strong motivation has been known to interfere with objective judgement.

Even in the absence of this effect, given the general lack of traditional wisdom available in such a new field, it is not surprising that many of the more common computer performance measures are, in fact, not well understood. The balance of this paper examines several common measures in some detail, shows how they are deceptive, and suggests straightforward alternatives which usually are no harder to apply.

Availability

One of the strengths of the English language is the ease with which old words can acquire new meanings. This facility can be a mixed blessing during the interregnum when the old and new meanings are both current. "Availability", as currently defined in the Computer Performance Measurement lexicon, is a case in point: simple transference of the old meaning would define "availability" as the fraction of the time the system can be used; whereas the new meaning is synonymous with "uptime": The fraction of scheduled time the system is available to the computer center.

The new meaning differs in three important aspects from the expected meaning:

(1) the time base is reduced, from time (meaning real time) to scheduled time; time devoted to such activities as preventive

maintenance or operating system development and installation is excluded from consideration;

(2) the time considered "available" includes many periods of time when the system is not, in fact, fully usable: time devoted to starting the system at the beginning of the day or after an interruption; time devoted to the "run down" before a scheduled interruption, when the input is shut off;

(3) the time considered "available" also usually includes time devoted to re-running lost or interrupted jobs.

"Availability" is a deceptive measure because these differences combine to give the impression of a much higher level of performance than that seen by the user. As an example, consider the week shown in Exhibit 1. We are assuming a one-shift operation (9-5, Monday-Friday), with preventive maintenance every Monday 9-10, systems development daily 1-2, and with run downs (input off) starting daily at 12:30 and 4:00. We will further assume that the system was down for 15 minutes (11:00-11:15) on Tuesday and one hour (10:00-11:00) on Friday, and that all starts and restarts require half an hour. The reported "availability" would be a seemingly respectable 96.3% (32.75 hours out of 34); the user, however, would find the system available (i.e., able to accept his input) only 48.1% (19.25 hours out of 40)! Discrepancies of this magnitude (which are not uncommon) help to explain the failure of some EDP managers to understand their users' cries of anguish and outrage.

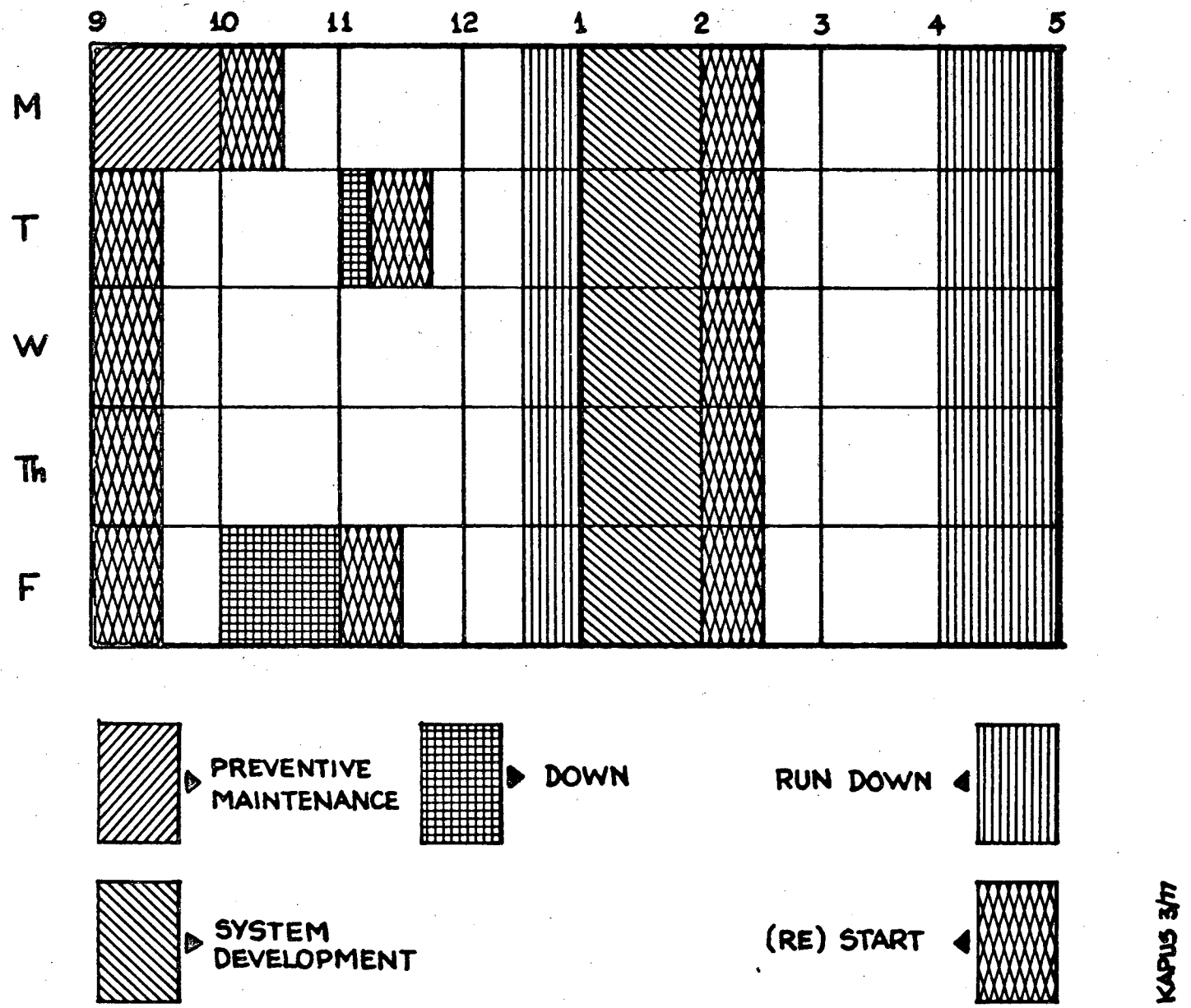

9
10
11
12
1
2
3
4
5
M
T
W
Th
F
PREVENTIVE MAINTENANCE
DOWN
RUN DOWN
SYSTEM DEVELOPMENT
(RE) START
KAPLUS 3/77

EXHIBIT 1.

There are two alternatives to this abuse of the term "availability". The simplest is to keep the measure and change its label to "uptime", but in view of the fact that it leaves even the EDP manager in ignorance of the true availability of his system, it is unsatisfactory. The proper approach is to keep the label, but to measure availability from the user's point of view; i.e., as a fraction of real time, as indicated in the example.

MTBI

"Mean time between interrupts" is a common measure of general system reliability. It can be derived for hardware (by considering only interrupts caused by hardware failure), for software, or for the total system. It is a generalization of a commonly used hardware reliability measure (mean time to failure, or MTF) and has little validity as a computing system reliability measure. It fails to recognize an essential difference between mechanical systems and computing systems: that failures in mechanical systems are caused by wear and fatigue, whereas failures in computing systems are caused by inadequate provision for unexpected input. (It is rare for an overflow on the meter to cause a taxi to crash; it is quite common for an equally trivial overflow to crash a multi-million-dollar computing system.) Increasing length of service is thus a much less significant source of failure than increasing diversity of workload.

The user, in any event, is more interested in the length of the service interval between his job input time and the next suspension of service, scheduled or not, than in MTBI. An accurate determination of this interval is beyond the scope of this paper, but an acceptable substitute is the mean service interval. To see how this compares with MTBI we return to the sample week of Exhibit 1.

As was noted above, the EDP manager would claim 32.75 hours of production, and would recognize 2 interruptions (the two down periods). MTBI is calculated by dividing the hours of service by the number of interruptions plus one: in this case, 32.75/3 or 10.9 hours...three times as long as the longest uninterrupted period of service seen by a user! The mean service interval, even giving full credit for the run-down periods, is 2.23 hours (26.75/12).

Although it is a less favorable measure (it will always be shorter than MTBI), the mean service interval corresponds much more closely to the user's view of the EDP facility than MTBI, and is therefore to be preferred.

Utilization

One problem with raw utilization measures is that they do not tell management by whom, for what, or how effectively the system was used. The situation has improved in recent years with the introduction of separate "system" and "problem" states for CPU utilization, but it remains the case that much of what is called "problem state" is actually system overhead. With respect to channel utilization, the situation is still primitive: channels can be "utilized" even when no data transfer is in progress, and there is no meter to differentiate between system (overhead) channel activity and user channel activity.

Another difficulty with utilization measures is that they reward ("encourage" may be too strong a word) inefficient usage: if one is being evaluated on the fullness of her basket she's not likely to spend very much time packing it more effectively.

I do not wish to give the impression that an EDP manager should not measure and know the level of utilization of the various devices under her control: quite the contrary, she should have that information at her fingertips at all times. I do wish to discourage the use of utilization measures in the evaluation of the EDP function by upper management. "Utilization" is a poor, and often counter-productive, approximation to throughput, the proper yardstick.

Unfortunately, there are no very good approximations, unless the workload is very stable. With a constant, periodic workload one can use jobs processed per period; with a strongly production-oriented shop one can use reports produced per period; some installations have defined "computing units"--combinations of CPU time and other resource usage charged to the user--with which they measure throughput. Many measures are possible; the ones selected should reflect the nature and the goals of the installation, and the computing power delivered to the user.

Overlap

The desirability of multiprogramming is now widely understood, but effective measures of the degree of multiprogramming are not so well known. "Overlap" as usually defined, whether CPU/Channel or Channel/Channel, indicates the existence of multiprogramming but does not measure its degree. The degree of multiprogramming is the number of simultaneous processes in operation; an acceptable approximation is the sum of the utilization of all channels plus the CPU. (This is generally not the same as the number of simultaneously active jobs or partitions.)

This measure shares the major disadvantage of all utilization measures: high system overhead and inefficient use contribute to a

"good" score. Nevertheless, it is a good indication of the capabilities of the system, and it should not be allowed to fall too far below the number of active partitions without investigation.

Efficiency

"Efficiency", like "availability", is a case of mis-labelling. When it appears in such contexts as "58% CPU efficiency" it always means utilization, whether efficient or not. (A program which adds two numbers together by repeatedly subtracting 1 from the first and adding it to the second can use 100% of the CPU, but it shouldn't be called 100% efficient.) The use of the word efficiency in this manner should not be tolerated...if you must talk of utilization, call it utilization.

Saturation; capacity

"Percent of saturation" is another misleading abuse of language. "Saturation" is not a measure but a binary condition: a system is or is not saturated; the difference is easy to detect: a saturated system is one which is being given more work to do than it can process under existing restraints, whether or not it's working to capacity, and the only reliable external manifestation is the lengthening input queue.

References to "80% of saturation" really mean 80% of capacity, and are doubly misleading because saturation is largely independent of capacity. We can best illustrate this by means of an example: Consider a long, narrow footbridge with a capacity of 2000 pounds. Thirty-nine small boys, each weighing less than 50 pounds, would not exhaust its capacity but would surely saturate it (they wouldn't all fit at once) for quite a while. A single man leading an elephant, on the other hand, would not saturate the bridge but would exceed its capacity.

Attempts to measure throughput as a percentage of capacity are misleading because they ignore the basic fact that "capacity" changes with workload and environment. It is well known that any reasonable multiprogramming system has less capacity when restricted to highly compute-bound jobs than when fed a mixture of compute- and I/O-bound work; it is less well understood that any multiprogramming system strongly dominated by priority considerations has less capacity than a system free to assign requested resources (such as the CPU) in an optimal fashion.[2]

Once more we find ourselves in the utilization-measure trap: an EDP manager evaluated on "production achieved as a percentage of capacity" is not strongly motivated to increase the capacity of his system, except by the addition of new equipment. It is clear that the proper measure is "maximum achievable capacity (with a given configuration)".

Lines of code

This measure as applied to programmer productivity is a repugnant outgrowth of early statistical studies. It is based upon the experiences of manufacturers, who are traditionally the producers of costly, bulky, inefficient, and hard-to-use programs. It addresses none of the qualitative aspects of computer programming. It rewards complex, voluminous, thoughtless code and penalizes thoughtful, elegant code; it is counter-productive. Programming is a creative process: was Horace evaluated on lines of ode?*

---

[2] Stevens, On overcoming high-priority paralysis in multiprogramming systems, CACM, August 1968.

* It is only fair to state at this point that the author began his data processing career as a programmer.

As the principles and practice of structured design become more widely used it becomes possible to consider the program module as an increment of measure. (Unfortunately, I cannot say unit of measure, because a unit should have a fixed size. Modules differ both in difficulty and in degree of success: a hard module is worth more than an easy one, and one which works and is simple, efficient, and easy to use is more successful than one which merely works.) If explicit measures of programmer productivity are necessary, a weighted count of modules should be used.

Interactive response time

"Interactive response time" is another meaningless statistical generalization. For response time measurements to be meaningful it is necessary to know response to what? under what conditions? at what time of day? The situation is further complicated by lack of definitive knowledge about what constitutes optimum response time: folklore exists to support each of the following views:

(a) fastest is best

(b) too fast causes anxiety and increases errors

(c) too slow causes impatience and increases errors

(d) most constant is best.

It seems clear* that optimal response contains elements of all four, and any measure which encourages one to the exclusion of the others is defective. (The standard "response time" measure encourages (a) to the exclusion of the rest.)

---

* This statement reflects the author's belief that that response is best which most closely matches the expectations of an optimistic user.

One way to alleviate the deficiencies of the standard measure is to classify the requests and examine the response time by classification. Responses to "trivial" requests (log-off; close a file;...), for instance, should be instantaneous; some requests (log on, say) may take a few seconds, but should instantaneously let the user know what's happening; others (complete searches of large, non-resident files) may justifiably require response times in excess of a minute. Another, partial, approach to the problem is to partition the response time into "process time" and "system wait time"; this, combined with the classification scheme suggested above, will provide the kind of information upon which one can base a rational evaluation of an interactive system.

Averages (means) and percentages

It is no coincidence that all of the specific misleading measures discussed above are either means or percentages: means and percentages are the easiest measures to obtain. One must be extremely careful in dealing with means and percentages, however, in order to ensure that they give an accurate and meaningful picture of the state of system. We have seen in the case of "availability" in particular how the selection of an artificially small base helps produce an artificially high percentage. And while averages are useful, they tend to obscure meaningful detail. Any time one uses averages, one should also, at least occasionally, examine the median and the distribution of the quantities averaged. As an example of the use of the distribution, let us consider the plot of mean service interval shown in Exhibit 2a. It seems clear that something bad started happening in August. In fact, as one can see from the distributions in Exhibits 2b and 2c, something good started happening in August: the expected service

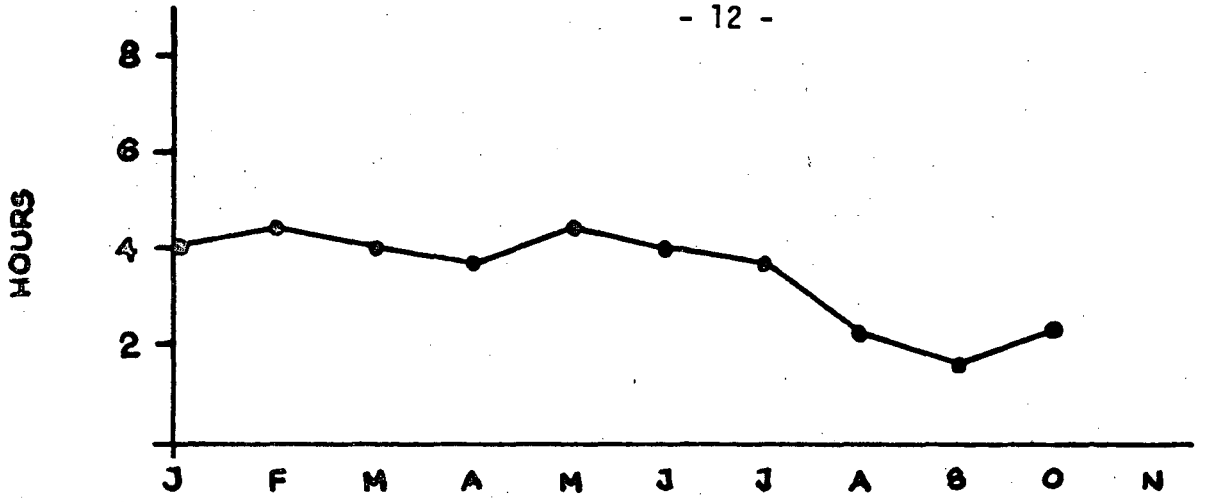


EXHIBIT 2a.

MEAN SERVICE INTERVAL

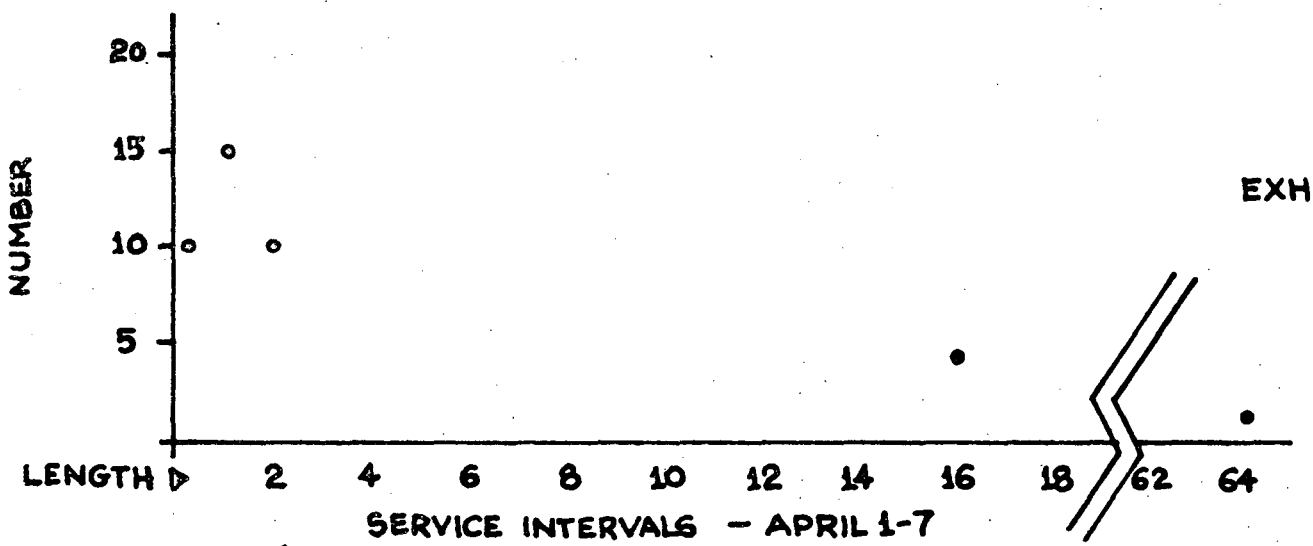


EXHIBIT 2b.

SERVICE INTERVALS – APRIL 1-7

(168 HOURS, 40 INTERVALS; MSI = 4.2 HRS.)

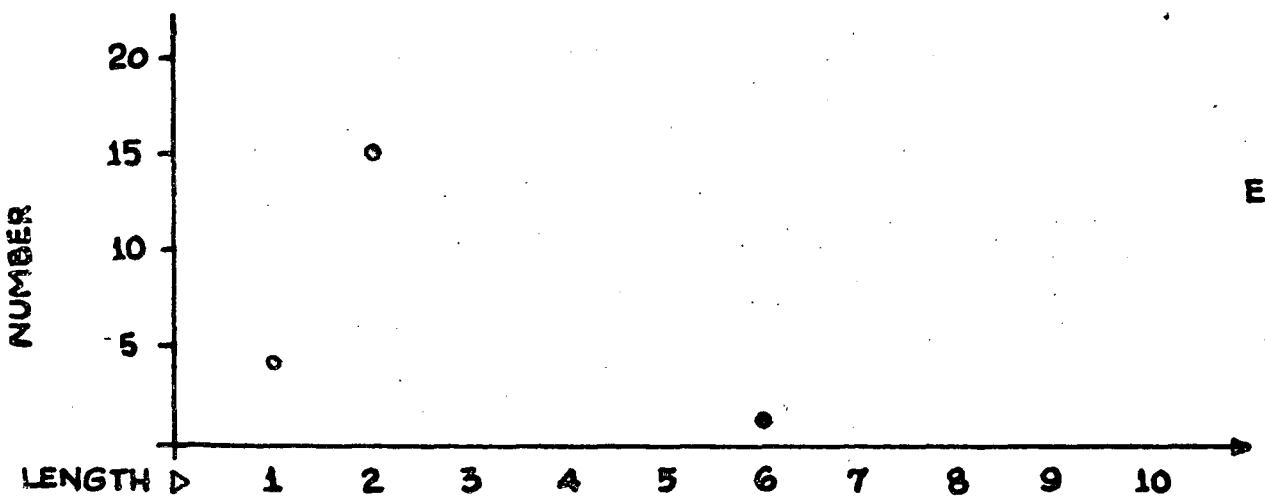


EXHIBIT 2c.

SERVICE INTERVALS – SEPT. 2-8

(40 HOURS, 20 INTERVALS; MSI = 2.0 HOURS)

interval during prime shift increased from about 1 hour to about 2 hours. The dip in the measured average was caused by the resulting possibility of eliminating the hitherto necessary, but relatively unused, off-shift rental hours.

It should also be clear from this example that one should not wait for an anomaly to appear before examining the distribution of a measured quantity: if the April distribution were not available in September it could not have been compared with the September distribution, and the determination of the true cause of the drop in the length of the average service interval would have been that much more difficult.

What does it all mean?

It means that upper management frequently doesn't know the true status of the EDP facility. Sometimes it means that the EDP manager himself doesn't know the true status of the EDP facility. It is recommended that the measures used to report EDP performance be reviewed; any which appear in the left-hand column of Exhibit 3 should be replaced with the appropriate measures from the right-hand column. It is further recommended that upper management frequently apply the twelve questions of Exhibit 4 to all performance measures employed. A continuing review of this sort is the only way to ensure that the performance reported is a reasonable facsimile of the performance perceived by the users.

Exhibit 3

MISLEADING MEASURES		RECOMMENDED HONEST MEASURES
Common name	What is actually measured	
Availability	Uptime	Availability for general-purpose use, as a percentage of a real time.
MTBI	Mean time to (hardware or software) crash	Mean Service Interval
Utilization	Resource occupancy	Throughput (work delivered)
Overlap	Existence of multi-programming	Degree of multiprogramming (the sum of utilization all channels, including CPU)
Efficiency	Resource occupancy	See "Utilization" above
Saturation	Work as % of capacity	Capacity; existence or not of saturation; throughput (in absolute terms)
Lines of code produced	(No one really knows)	Working modules produced (weighted for quality and difficulty)
Interactive response time	(Nothing)	Process time vs. system wait time, possibly by category

Exhibit 4

Review questions for performance measures

A. For all measures

(1) What does it tell me?

(2) How does it do that?

(3) Why is that useful?

(4) Does it agree with the experience of the users?

(5) What is the trend of the measurement (i.e., is it larger or smaller than before)?

(6) Is that significant?

B. Additional questions for percentages

(7) Percentage of what?

(8) Is that the most reasonable base?

C. Additional questions for averages

(9) Why use average instead of median?

(10) What is the median?

(11) What is the difference between average and median and what does it signify?

(12) What is the distribution?

This report was done with support from the United States Energy Research and Development Administration. Any conclusions or opinions expressed in this report represent solely those of the author(s) and not necessarily those of The Regents of the University of California, the Lawrence Berkeley Laboratory or the United States Energy Research and Development Administration.

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