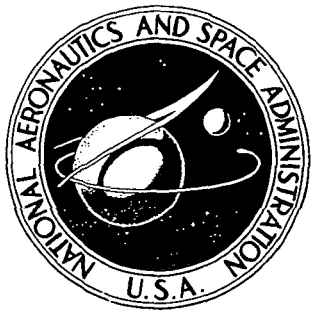
NASA TECHNICAL NOTE



NASA TN D-6873

NASA TN D-6873

A WORKLOAD MODEL AND MEASURES FOR COMPUTER PERFORMANCE EVALUATION

by H. Kerner and K. Kuemmerle

George C. Marshall Space Flight Center
Marshall Space Flight Center, Ala. 35812

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • OCTOBER 1972

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TN D-6873		

4. Title and Subtitle	5. Report Date
A Workload Model and Measures for Computer Performance Evaluation	October 1972
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
H. Kerner and K. Kuemmerle	
	10. Work Unit No.

9. Performing Organization Name and Address	11. Contract or Grant No.
George C. Marshall Space Flight Center Marshall Space Flight Center, Alabama 35812	
	13. Type of Report and Period Covered

12. Sponsoring Agency Name and Address	
National Aeronautics and Space Administration Washington, D. C. 20546	Technical Note
	14. Sponsoring Agency Code

15. Supplementary Notes

Prepared by Computation Laboratory, Science and Engineering

16. Abstract

A generalized workload definition is presented which constructs measurable workloads of unit size from workload elements, called Elementary Processes. An Elementary Process makes almost exclusive use of one of the processors, CPU, I/O processor, etc., and is measured by the cost of its execution. Various kinds of user programs can be simulated by quantitative composition of Elementary Processes into a Type. The character of the Type is defined by the weights of its Elementary Processes and its structure by the amount and sequence of transitions between its Elementary Processes. A set of Types is batched to a Mix. Mixes of identical cost are considered as equivalent amounts of workload. These formalized descriptions of workloads allow investigators to compare the results of different studies quantitatively. Since workloads of different composition are assigned a unit of cost, these descriptions enable determination of cost effectiveness of different workloads on a machine. Subsequently performance parameters such as Throughput Rate, Gain Factor, Internal and External Delay Factors are defined and used to demonstrate the effects of various workload attributes on the performance of a selected large scale computer system.

17. Key Words (Suggested by Author(s))	18. Distribution Statement

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	30	$3.00

* For sale by the National Technical Information Service, Springfield, Virginia 22151

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

A WORKLOAD MODEL AND MEASURES FOR COMPUTER PERFORMANCE EVALUATION

INTRODUCTION

Performance evaluation of large computer systems has developed into a branch of computer science. Initially, comparisons of memory cycle times or instruction execution times sufficed. However, evaluation became much more difficult when some instruction executions could overlap in time, such as arithmetic or logic operations and input/output actions on one or more channels. Other forms of overlap in time can be seen in interleaved accesses to data and instructions. Through the exploitations of such overlaps, the system's software can enhance the performance of computer hardware considerably. Since a large population of programs provides a greater opportunity for occurrence of simultaneously executable instructions than a single program, multiprogramming and multiprocessing of programs were introduced. Evaluating the success of such complex architectural designs became an important factor in both the design of computers and the selection of a computer or a configuration of a system.

Performance analysis deals with three factors: the architecture of a system, workload to be processed, and cost structure of the system. Architecture shall be understood in a broad sense as all elements, other than the user programs, influencing the performance of a system. Besides the hardware structure and configuration of a machine, it includes the total system software consisting of the operating system, compilers, library routines, utility programs, etc. It also encompasses the installation's policies concerning the operation of the machine (e.g., to run only short jobs during daytime and longer jobs for night shifts, etc.) and programming policies (e.g., policies regarding the use of faster or slower drums, of storage of recurring jobs, etc.). Because performance depends not only on the type of machine but also on an associated workload, a generalized description of this workload is required. The representation of a workload in published performance studies varies widely with the technique employed. Mathematical models represent their workload by the probability distributions (4), (5), and (7). Simulation studies of computer performance use either a statistical description by distribution functions or a deterministic description of the workload [equations (6) and (9)]. Experimental investigations of existing computers use benchmarks representing typical selected workloads [equations (2) and (3)].

This report treats performance measurement by experiment. The description of a workload by benchmark jobs restricts the observation of performance to a few particular cases, however representative for a category of programs; the workload representation proposed in this report describes a wide variety of programs and allows a gradual transition from one category to another; e.g., from compute bound jobs to I/O bound jobs. Most importantly, it allows a quantitative comparison of different workloads. In addition, the workload representation allows the structure of its constituent programs to be described in a manner such that their influence as individual programs on the utilization of the subsystems becomes highly visible. Further, this visibility is preserved when the individual programs are combined to form a Mix. The workload so constructed is entirely portable. This report attempts to satisfy these criteria by a generalized description of the workload.

Since an expensive machine is expected to perform better than a low cost facility, and especially since each of its subsystems (CPU, drum, disc, tape) influences the performance, the machine will be identified by cost factors, which will be defined during the introduction of the workload description. The cost structure of a machine is used in defining the quantity of a workload. Workloads will be considered as equivalent on one machine if their serial execution of the different subsystems used costs the same amount.

Different users will define performance in different ways. One group considers mainly the total number of useful actions performed by the system in a time period of 24 hours, another group is concerned with the time taken by the system in responding to its request, and a third group is concerned with the cost effectiveness of the system as measured by the utilization of its subsystems. One can, therefore, see the need for at least three different system performance measures, which will be defined together with some subsystem utilization measures in a later section. To demonstrate the utility of the definitions of the workloads and the performance measures, several problem examples are given in the section entitled Applications.

The workload and measures described in this report should also be useful for simulation studies and provide the link to mathematical models, which will ultimately need to be verified by comparison with results obtained from experiments. It is hoped that the proposed workload attributes provide a basis of sufficient generality to allow investigation of widely differing architectures and workloads.

GENERALIZED WORKLOAD REPRESENTATION

The workload is defined in three levels: Elementary Processes, Types, and Mixes, where Types are combinations of Elementary Processes equivalent to individual programs (jobs) and Mixes combine Types into the substitute for a batch of programs.

Elementary Processes

An Elementary Process is a very short program used as a standard which mainly uses only one of the processors of a machine, CPU's and I/O channels/controllers, but can be supported to a minor degree by other processors. For the purpose of this report, the Elementary Processes are those defined in Table 1.

TABLE 1. ELEMENTARY PROCESSES

PA	Central Process	A
PH	Central Process	H
R	Random Access Process	
S	Serial Access Process	
C	Card Read Process	
PR	Print Access Process	

The PA Elementary Process consists predominantly of floating point arithmetic operations, while the PH Process contains mainly fixed point arithmetic, logical, and housekeeping instructions in the CPU. The Random Access Process is determined through a program that transfers records or blocks of records of randomly ordered files from disc or from drum to core and vice versa, while the Serial Access Process deals only with sequentially ordered files. The two other processes are self-explanatory.

The programs representing the Elementary Processes constitute a set of standards for measuring a unit of workload. In order to compare the

performance of different workloads on a given system with constant architecture and cost structure, it is necessary to introduce a metric for the Elementary Processes. A unit of an Elementary Process will be represented by its associated program, which is adjusted to execute for the length of time paid by the unit of cost on the processor under consideration. It is obvious that a unit of an Elementary Process is machine-dependent because it is a function of the cost structure of the system. A unit of an Elementary Process, therefore, lasts longer on a low cost device. This sizing of workloads according to the cost of execution of their Elementary Processes allows the comparison of different workloads on a machine and the evaluation of the cost efficiency of a machine. If two different architectures are to be compared, each of the Elementary Processes can be normalized to execute for a unit of cost and the amount of work executed can be compared by counting the number of iterations of an Elementary Process on each of the machines.

As mentioned above, an Elementary Process should, ideally, be a program that uses one of the physical processors of the system exclusively; these physical processors are CPU's and I/O channels connected with some devices. In reality, however, it is not feasible to have a one-to-one correspondence between the Elementary Process and the physical processor; e.g., a Random Access Process needs some minor support by the CPU. Since it is necessary to establish a correspondence between the Elementary Process and the associated physical processors for determining its cost, a Virtual Elementary Processor is defined as the aggregate of physical processors involved in the execution of an Elementary Process. The cost of the Virtual Elementary Processor is composed of the costs of its participant physical processors in proportion to their utilization during the execution of the respective Elementary Process; e.g., the cost for a Virtual Random Access Processor is the sum of the costs of the respective device controller, the I/O processor (channel), and a (small) CPU involvement. The costs are known from manufacturer's price lists, and the amount of participation can be measured; for example, using software or hardware monitors. The determination of the costs of Virtual Elementary Processes will be further illuminated by discussing the following cases:

1. Two, or more, Elementary Processes may use the same physical processor. An example is the utilization of one type of CPU, possibly realized as a set of identical CPU's, by two different Elementary Processes, such as the PA (floating point arithmetic) and the PH Processes (housekeeping). In this case the Virtual Processors are not composed of physical processors, but obtained by partitioning a physical processor. Therefore, the cost information cannot be obtained from price lists but must be estimated.

2. An Elementary Process may have a choice between different physical devices for its execution; for example, a Random Access Process may be executed either on a fast drum or on a slow disc device. Hence, several Virtual Elementary Processors are available to be chosen for the execution of one Elementary Process, each with a different participation of physical devices and with a different resulting cost. The derivation of this cost, however, follows the usual scheme.

3. If there is not a choice between the different Virtual Processors and several Virtual Processors are used concurrently, one Elementary Process must be introduced for each Virtual Processor employed.

For comparisons of some workloads on different machines, one machine will be considered as the reference machine with its cost structure used for the definition of units of the Elementary Processes. For this type of investigation, it is possible that some Elementary Processes defined on the reference machine cannot be executed on a machine to be compared; e.g., a Random Access Process defined on the reference machine cannot be executed on a machine devoid of random access devices. This problem can be solved by substituting a combination of other Elementary Processes for the problematic Elementary Process. For example, a combination of Sequential Access Processes and PH Processes can be substituted for a nonexecutable Random Access Process.

To show the relationship between Elementary Processes and the cost of corresponding Virtual Processors, the list of Elementary Processes is now expanded by the cost factors associated with their respective Virtual Elementary Processor and presented in Table 2. The numerical example applies to a UNIVAC 1108 with three CPU's and two ICO's. Note that some Elementary Processes have multiple cost factors associated with them, indicating a choice of several Virtual Elementary Processes. From now on the set of Virtual Processor cost factors will be referred to as the cost vector of a system,

$$CP = (Cpa, Cph, Crl, \ldots, Cc) \quad .$$

The elements of this vector have the dimension dollars per hour ($/h).

Each Elementary Process requests a certain storage area in one or several storage media. The system's performance may very well depend on its ability to satisfy the storage requirements of a set of programs. To

TABLE 2. VIRTUAL ELEMENTARY PROCESSORS

Elementary Process	Virtual Processor Cost Factor	Dollars per Second	Processor Used
Central Process A	Cpa	$5.4 \cdot 10^{-3}$	CPU
Central Process H	Cph	$5.4 \cdot 10^{-3}$	CPU
Random Access Process	Cr1	$0.97 \cdot 10^{-3}$	Fast Drum Channel
Random Access Process	Cr2	$0.78 \cdot 10^{-3}$	Slow Drum Channel
Random Access Process	Cr3	$0.97 \cdot 10^{-3}$	Disc Channel
Sequential Access Process	Cs1	$0.80 \cdot 10^{-3}$	Fast Tape Channel
Sequential Access Process	Cs2		Slow Tape Channel
Print Process	Cpr		Printer Channel
Card Read Process	Cc		Card Reader Channel

account for this, we define a set of Elementary Storage Requirements which indicate the amount of storage needed for the execution of the Elementary Process (core storage, discs, drum tracks, or tape drives). The Elementary Storage Requirements are measured in kilobits (kb) or by the number of tape drives. A Storage Cost Factor is associated with each of the Elementary Storage Requirements. It is measured in $/bit * hours and $/tape drive hours. The Elementary Storage Requirements and the corresponding Storage Cost Factors are represented in Table 3 for the example of a Random Access Process.

TABLE 3. STORAGE REQUIREMENT AND COST EXAMPLE FOR A HYPOTHETICAL RANDOM ACCESS ELEMENTARY PROCESS

Storage	Elementary Storage Requirement in kb	Storage Cost, $/kb * hour
Core	36	$0.56*10^{-3}$
RAS Drum A		$0.02*10^{-3}$
Drum B		$0.14*10^{-3}$
Disc	1000	$0.004*10^{-3}$
SAS Tape A		Cost of drive/hour
Tape B		Cost of drive/hour

The storage requirements can be considered as elements of a storage vector, s. The costs for the devices are measured in $/bits * hours and combined into a storage cost vector, CS.

Types

After the definitions of the Elementary Processes and Elementary Storage Requirements, it is now possible to proceed to an abstract description of a program (job) as a well defined combination of the Elementary Processes

which will be called Type. The Type receives its character from the relative weight of each of the Elementary Processes in it, the accumulative Elementary Storage required for all of the Elementary Processes present in the Type, and the sequence and frequency of transitions between the Elementary Processes.

To define a Type quantitatively, we first have to recall that a unit of an Elementary Process was characterized as an Elementary Process adjusted in such a way that its execution cost was unit cost. Now each type can be normalized to a Type of unit cost, a vector T, in which the elements represent the relative weights of the different units of Elementary Processes constituting the Type:

$$T = (Wpa, Wph, \ldots, Wpr) \quad , \qquad (1)$$

where

$$\sum Wi = 1 \quad .$$

The weight vector for a Type can be interpreted as a vector in the n-dimensional space, with each Elementary Process occupying one dimension. Figure 1 depicts an Elementary Process space reduced to three dimensions. The components of the type vector (T) are the weights of its Elementary Processes. Since the sum of the weights must equal one, each type vector must terminate in the hyper-plane expressed by equation (1).

It should be noted, that a Type will not only consist of the Elementary Processes defined above, but will also need a driver to sequence their execution according to the transition matrix which is still to be described. The driver will perform only housekeeping functions. It can, therefore, be considered as part of the Elementary Process PH. Hence, the PH Process load content of a program to be represented by a Type must be reduced in order to include the driver, otherwise the weight W_{12} will indicate the inaccuracy introduced by the driver. For example, a Type of unit cost may be represented by the weight vector W in Table 4; the driver contributes the weight Wph of the PH process. If the Type runs until it consumes \$100, it spends \$30 for the PA Processor, \$3 for the PH Processor (e.g., for the driver), etc.

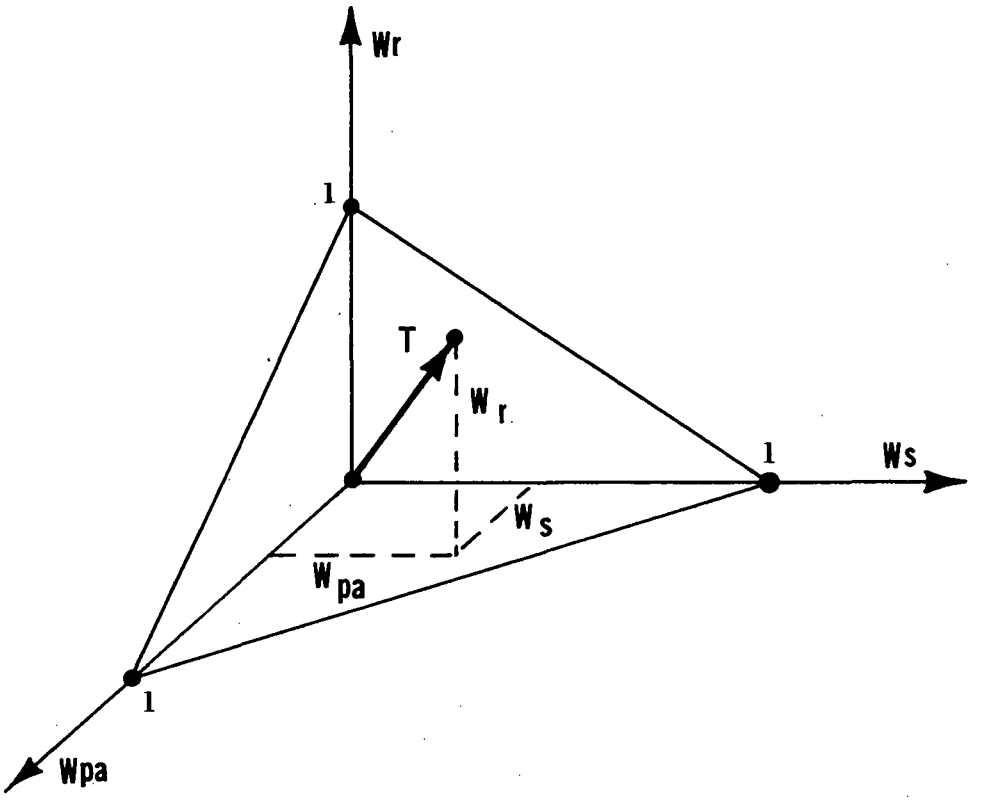


Figure 1. Elementary Process space reduced to three dimensions.

Table 4. Weight Vector W

	Central Process		Random Access		Sequential	Print	Card
W	Wpa	Wph	Wrl	Wr2	Ws	Wc	Wpr
1	0.3	0.03	0.1	0.3	0.2	0.04	0.03

The Storage Requirement for a Type is simply the sum of the storage required for all the constituent Elementary Processes. The storage vectors for each Elementary Process of the previous example are shown in Table 5. Except for the central processing functions PA and PH, all other Elementary Processes require two kinds of storage; namely, core and some mass storage. Often two Elementary Processes run cooperatively, sharing some storage areas. In the examples presented here a cooperation was assumed between R1 and R2 for transferring data from disc to drum. A core area of 4 kb,

TABLE 5. STORAGE REQUIREMENT FOR A TYPE

Elementary Process	Core kb	Drum kb	Disc kb	Tape Drives
PA	20	-	-	-
PH	1	-	-	-
R1	4	200	-	-
R2	2, R1	-	400	-
S	R2	-	R2	-
PP	0.5	4	-	-
C	R2	-	R2	-
Type	28.7	204	400	-
Storage in kilobits				No. Tape Drives

which is used as a buffer, must be reserved only once for these cooperative processes. In a previous example, the Process R1 accounts for this core buffer, while Process R2 refers only to it by listing the cooperative Process R1. We also recognize a cooperation between S and R2 (tape to disc transfer) involving a 2 kb buffer and observe a card to disc process (C-R2) sharing this 2 kb core buffer and the shared drum storage of 400 kb, which also serves in the S-R2 transfer.

The last line in Table 5 shows the storage requirement vector S for the Type which is scaled in kilobits and number of tape drives. The scalar product between the storage vector S and the storage cost vector CS for this machine results in the storage costs per hour for this Type. If all storage in the machine to be measured is reserved for the duration of a job, the storage cost per hour must be multiplied by the basic execution time required for the execution of the Type. The basic execution time of a Type will be defined as the time needed for a serial, not overlapping, execution of its constituent Elementary Processes. In this way, a cost for storage, expressed in dollars, will

result giving a measure of the total storage requirement for the Type. The performance of a program on a machine will not only depend on its Elementary Processes and Storage Requirements, but also on the sequence in which the processes are executed and on the degree of hashing between processes. A definition of these two features will therefore be added to the description of a Type.

Sequencing and hashing may occur in various ways: A Type given by a weight and storage vector may execute each of its Elementary Processes for the total time prescribed by its weight, before proceeding to the next process; or in the general case, it may execute only for a portion of the total time allocated for each of its Elementary Processes and return several times through a prescribed sequence, until each Elementary Process is executed for the full time prescribed for it. The sequencing of the Elementary Processes can be described by a transition table, as exemplified by Table 6 in which each Elementary Process is completed before switching to the next one. The first quantity of the pair of entries indicates the time spent for the Processes listed in the column, and the second one points to the Elementary Process to be executed next. The sequence starts with the Read Process (C) for the time t_4, then the Tape Process (S) is executed for the time t_3, this is followed by the execution of CPU-Process PA for the time t_{11}, etc. A Type with the same weight/storage vector can, however, be executed in a different manner by hashing its Elementary Processes. In this case the Transition Table would show loops between Elementary Processes whereby the frequency of iteration within a loop is expressed by an exponent for the time/transition pair. Whenever the performance of a system does not depend on the sequence of Elementary Processes, the workload can be presented in a much simpler form by merely listing the duration (t) and hashing frequency (n) of each Elementary Process, as shown in the bottom row of Table 6.

If the Type is not artificially constructed, as it was in this report, but is derived from a batch of jobs representing a typical day or hour on a machine, it is possible and desirable to compute the standard deviations, in addition to the averages of durations for each Elementary Process. While the authors refrain from using the standard deviation for the construction of a Type, it may contribute to the construction of a model of the total batch of programs. The curve in Figure 2 indicates the description of Elementary Process durations as Gaussian distributions.

Conceivably, the workload throughput of some architectures may depend on a blocking of Elementary Processes; e.g., in executing a block of concurrently executable Elementary Processes such as an "R2-PH" pair, the time needed for repeated initiations of these Elementary Processes may be eliminated. Also, the opposite effect may occur; larger blocks may hinder concurrent

TABLE 6. TRANSITION TABLE

Step	PA	PH	R1	R2	S	C	PR
START 1						t_4, S	
2					t_3, PA		
3	t_{11}, R1						
4			t_{21}, PH				
5		t_{12}, R2					
6				t_{22}, PR			
7							t_5, END
Per Elementary Process	t_{11}	t_{12}	t_{21}	t_{22}	t_3	t_4	t_5
Total Time	n_{11}	n_{12}	n_{21}	n_{22}	n_3	n_4	n_5

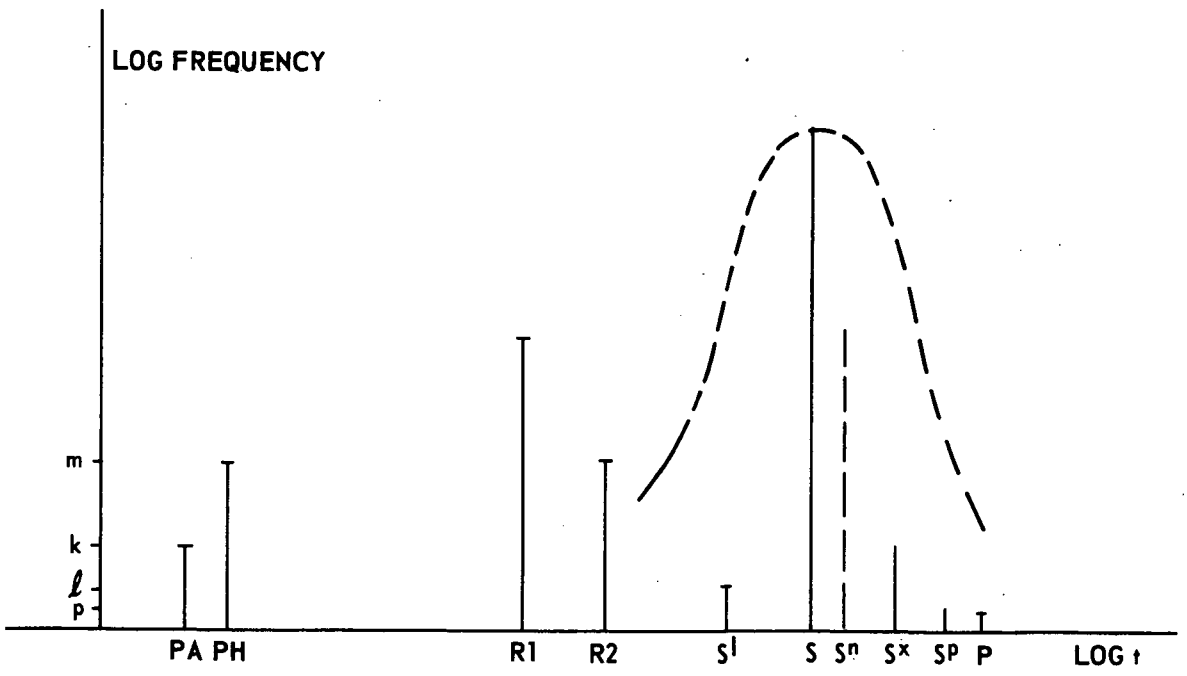


Figure 2. Frequency of Elementary Process durations.

processing of many different Elementary Processes. A Type description should therefore allow an architecture to fully display its management abilities.

All information on blocking of the processes is employed in Table 7. The value of each element of this matrix denotes the number of transitions from the Elementary Process, listed in the top row, to the Process named in the right column; e.g., there is a block of p consequentive S Processes, furthermore, there is one block of m loops between the R2 and PA Processes and a block of r loops through the trio of the Elementary Processes PH-R2-S, etc.

The Type specification could be further sophisticated to include the sequencing through the Elementary Processes by prescribing a random walk through the Processes. Each step toward refinement, however, creates an added burden for the extraction of such additional information from a typical

TABLE 7. BLOCKING OF TRANSITIONS BETWEEN ELEMENTARY PROCESSES

From Elementary Process							
PA	PH	R1	R2	S	C	PR	To
0		1	m				PA
1	0			k-1 r			PH
		0		n			R1
m-1	r		0				R2
	k	n-1	r	p	1	ℓ-1	S
					0		C
				ℓ 1		0	PR

batch of jobs and for the comparison of different workloads. For most architectures, a Type specified by the frequency and average duration of each of its Elementary Processes seems to be adequate.

Mixes

Multiprogramming systems execute portions of different programs concurrently whenever the appropriate processors are available. To test such systems, a large number of jobs must be simultaneously offered for execution. Accordingly, a large number of Types, batched into a Mix, are used as a proper workload. Originally, one may think of a Mix as a number of different Types of various lengths of execution, each Type to be executed repeatedly for a given number of times. Obviously, one can also construct a new Type which represents the average of the Types in the Mix.

The average Type is obtained by weighting each Elementary Process by both its original weight in a Type and the execution length and number of repetitions of this Type within the Mix. The new homogenized Type will be called the Mix Average Type, which has naturally all the properties of a Type, particularly that of unit cost. It can be specified by any one of the methods applicable for Type definitions. For simplicity, a specification using only the frequencies and average durations of the contributing Elementary Processes is preferred. Similarly, an average Storage Requirement for the Mix Average Type can be constructed.

The Mix consists finally of an infinite number of Mix Average Types, possibly with Elementary Process durations obtained from sampling prescribed Process distributions. The number of Mix Types simultaneously admitted to the system for execution will be determined by the Storage Requirement for the Mix Type and the system's storage capabilities.

Review of Workload Specifications

In review, we see the Elementary Processes as standards defining the units of processing. Execution times of these Elementary Processes are determined by the cost structure of the machine; the frequency of iterations reflect the differences in performance of the instruction sets available on the machine under test. Types are constructed to represent jobs. A gain in the execution time of a type compared with the time needed to execute its Elementary Processes serially reflects the machine's capability to overlap Elementary Processes.

Similarly, a gain in execution of a Mix over the time needed to process its Types serially indicates how well the architecture takes advantage of the simultaneous offering of a large number of Processes present in a Mix.

A workload represented in generalized form exhibits several properties: It allows a quantitative comparison of different workloads, it can be derived from benchmarks (either manually or supported by software monitors), it establishes a link between benchmark representations and statistical distributions of a workload, and it is portable between different machines. While the workload has only been described for batch processes, it can be expanded into real time and data base systems by adding timing or data access conditions to the transition table of a Type or by introducing such conditions into a Mix.

DEFINITION OF PERFORMANCE AND EFFICIENCY MEASURES

As previously stated, different users consider different features of a system important for them. In this chapter, parameters will be defined which describe the overall performance of a system rather than focusing on detailed features. No effort has been made to accommodate the special needs of real-time or time-sharing systems. In order to define parameters of performance and efficiency we will first describe the main check points through which a Type passes.

Throughput Rate, THR

As a measure of the performance of a data processing system one usually counts the number of jobs or instructions pertaining to a given workload processed per unit time. The following redefinition of this measure which is called the Throughput Rate, THR, in essence arose from this concept. In multiprogramming systems the active system time $ast_i = tt_i - at_i$ (Fig. 3) of a certain Type does not depend only on its structure and the processing speed of the system but also on the other Types being active at the same time and sharing core, CPU, I/O channels, and peripheral systems with it. The active system times ast_i of the Types constituting a Mix can be considered as realizations of a random variable, AST, with the expectation E [AST]. Its distribution function needs not to be specified for our purposes. If, on the average, r Types are active simultaneously (degree of multiprogramming) the Throughput Rate, THR, or in other words the Type terminating rate, can be defined as

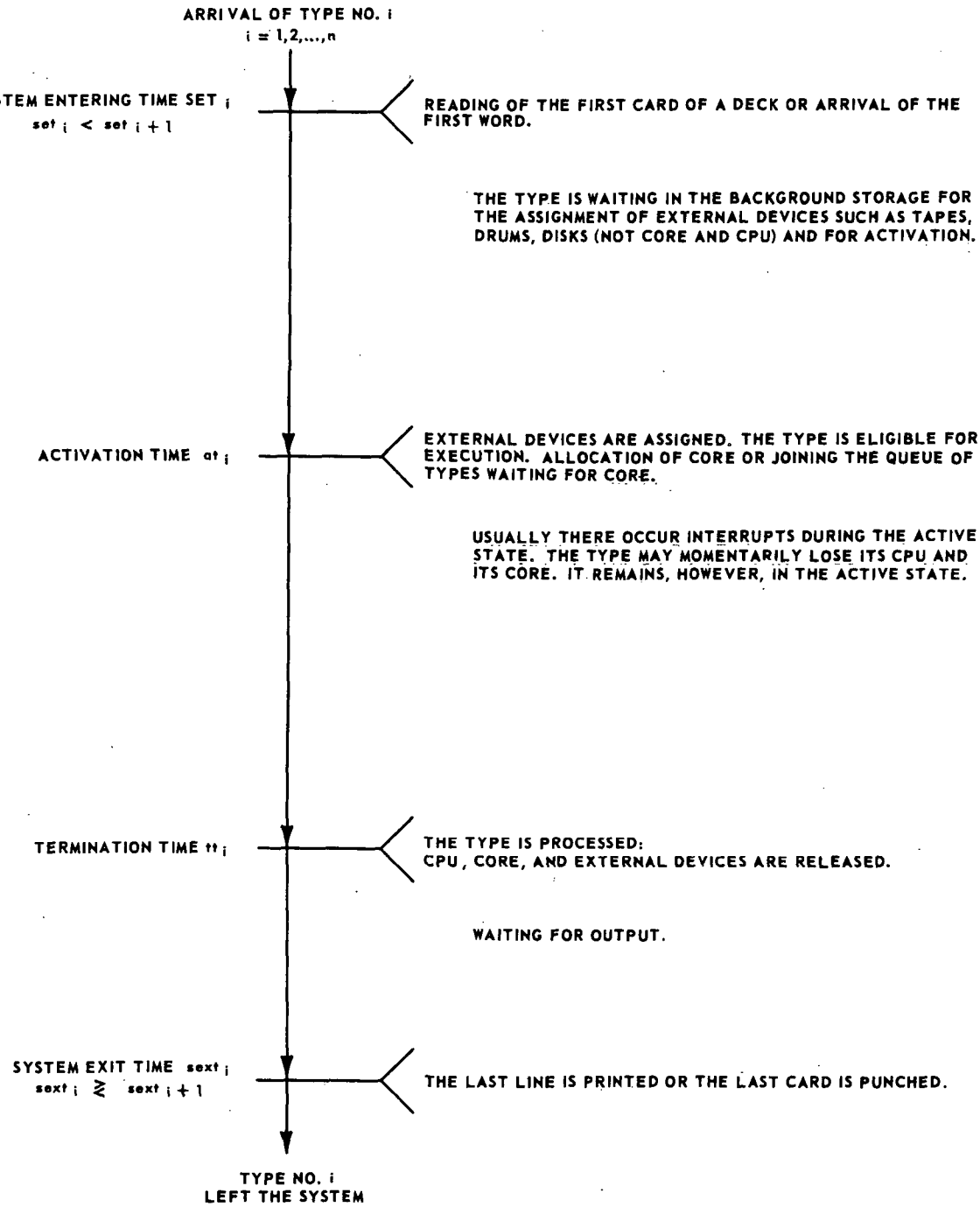


Figure 3. Active system time.

$$THR = \frac{r}{E\,[AST]} \quad . \qquad (2a)$$

This definition can be used for mathematical models and for measurements as well. In the case of measurements the following equation holds:

$$THR = \frac{n}{\max(tt_i) - at_1} \quad i = 1, 2, \ldots, n \quad , \qquad (2b)$$

where n is the number of Types processed during the total observation time $\max(tt_i) - at_1$.

Gain Factor, GF

It is well known that in a multiprogramming/multiprocessing system the total time elapsed for processing n Types of a workload is usually less than the time needed for serial processing of the same Types. More precisely, the interval of the total active time $\max(tt_i) - at_1$ measured on a multiprogramming/multiprocessing system is usually less than the corresponding time interval $tt_n^* - at_1$ for serial processing. The Gain Factor, GF, indicates how many times faster a system can process a given workload using multiprogramming compared to uniprogramming.

$$GF = \frac{tt_n^* - at_1}{\max(tt_i) - at_1} \quad . \qquad (3)$$

Comment: The combination of equation (3) with equation (2b) yields:

$$GF = THR * \frac{tt_n^* - at_1}{n} \quad . \qquad (4)$$

This equation suggests another physical interpretation of the Gain Factor, GF. Since the second term in equation (4) represents the mean active system time

per Type for serial processing, the quantity GF can be interpreted as the mean number of Types processed during that time. Therefore, according to the notation Throughput Rate, THR, one could use the term Throughput instead of Gain Factor.

Internal Delay Factor, IDF

Despite the gain in the total time required to process a batch of Types in a multiprogramming environment, the active system times for the individual Types will be greater than the corresponding times for serial processing. The increased active system times of individual Types is the price paid for a better utilization of all processors of a computer system. To measure this increase in active system time, an Internal Delay Factor, IDF, is introduced. This indicates how many times greater the expectation E [AST] of the active system time per Type is with respect to the mean active system time

$\frac{1}{n}\,(tt_n^* - at_1)$ for serial processing:

$$IDF = \frac{E\,[AST]}{\frac{1}{n}\,(tt_n^* - at_1)} \quad . \tag{5a}$$

If the Internal Delay Factor is determined by measurements, the expectation E [AST] has to be replaced by the corresponding mean sample value:

$$IDF = \frac{\sum_{i=1}^{n} (tt_i - at_i)}{tt_n^* - at_1} \quad . \tag{5b}$$

Comment: According to equations (3) and (5b) there exists a relationship between the Gain Factor, GF, and the Internal Delay Factor, IDF:

$$GF \cdot IDF = \frac{\sum_{i=1}^{n} (tt_i - at_i)}{\max(tt_i) - at_1} \quad . \tag{6}$$

The right side of equation (6) represents the mean number of jobs which are active simultaneously.

External Delay Factor, EDF

Usually the intervals between the system entering time, set_i, the activation time, at_i, between the termination time, tt_i, and the system exit time, $sext_i$, respectively, are significantly different. Therefore, it seems to be appropriate to introduce another characteristic quantity, the External Delay Factor, EDF. If we consider the total system times, $tst_i = sext_i - set_i$, of the Types constituting a Mix as realizations of the random variable TST, then the External Delay Factor indicates how many times the expectation E [TST] of the total system time per Type is greater than the expectation E[AST] of the active system time per Type under the same operating conditions.

$$EDF = \frac{E\,[TST]}{E\,[AST]} \qquad (7a)$$

When EDF has to be determined by measurements the expectation values in equation (7a) have to be replaced by the corresponding mean sample values:

$$EDF = \frac{\sum_{i=1}^{n} (sext_i - set_i)}{\sum_{i=1}^{n} (tt_i - at_i)} \qquad (7b)$$

Utilization Factors

The parameters defined previously do not reflect the utilization of the main components of a system such as CPU's, I/O channels, and working storage. Although, the utilization factors of these components are not performance parameters proper, they provide at least some insight into how the workload utilizes the system's capacity. Very low utilization factors can indicate excess system capacity, a mismatch of the workload with respect to the system's architecture, an inefficient algorithm of the operating system,

or that the degree of multiprogramming should be increased. Very high utilization factors of I/O channels, however, point out excessively long queues and waiting times. Subsequently, several utilization factors will be defined.

- Utilization of the Central Processing Unit

  1. Utilization of CPU no. j; $j = 1, 2, \ldots, m$

where

$$UCPUj = \frac{\text{Active time of CPU no. j}}{\max(tt_i) - at_1} \quad . \tag{8a}$$

  2. Overall CPU utilization

$$UCPU = \frac{1}{m} \sum_{j=1}^{m} UCPUj \quad . \tag{8b}$$

Comment: The active time of a CPU is supposed to include both the CPU time charged to the individual Types of a Mix and the CPU time needed for performing housekeeping functions (system overhead SOH). Although the system overhead is not considered to be a performance parameter proper, it should be considered together with the CPU utilizations,

$$SOH = \frac{\sum_{j=1}^{m} \text{active time CPU no. j} - \sum_{i=1}^{n} \text{CPU time charged to Type no. i}}{\sum_{j=1}^{m} \text{active time CPU no. j}} \quad . \tag{8c}$$

- Utilization of I/O Channels

  3. Utilization of channel no. k; $k = 1, 2, \ldots, \ell$

$$UCH_k = \frac{\text{Busy time channel no. k}}{\max(tt_i) - at_1} \quad . \tag{9a}$$

4. Overall channel utilization,

$$UCH = \frac{1}{\ell} \sum_{k=1}^{\ell} UCH_k \quad . \tag{9b}$$

- Utilization of Core Memory

Working storage (core) is usually required for three purposes: storage of the resident part of the executive system, storage of active Types and nonresident parts of the executive system, and as buffers. In this report, only the utilization of that part of core memory which is available for Types (user programs) and nonresident routines and functions of the executive system is considered. Its capacity is denoted by C. To determine the utilization of C the portion $C_{mom,t}$ occupied momentarily at sample time t is measured. If we denote the time interval between two consecutive sample points by Δt, the utilization of core memory is

$$UCM = \frac{\sum_{t} C_{mom,t} * \Delta t}{C * \max(tt_i - at_1)} \quad . \tag{10}$$

In order to obtain numerical values for the parameters defined above, the quantities specified in equations (2b) through (10) have to be measured. There are three different sources of information: the accounting system of a computer installation, software monitors, and hardware monitors. Because it is the authors' intention to define performance and efficiency parameters rather than to focus on measurement techniques, these data acquisition methods will not be discussed.

APPLICATIONS

The usefulness of the definitions and approaches outlined in the previous sections will be demonstrated by applying them to performance studies of a large scale computer, the UNIVAC 1108, a system employing three central processing units and two I/O control units operating under the executive system EXEC 8. Four classes of problems are treated.

First, the influence of the type of Workload on the Performance of a constant architecture is investigated. Such investigations become feasible because of the assignment of a value, the cost of executions to each Type. We recall that the character of a workload is represented by the weights of the Elementary Processes present in the Mix Type. To study the global behavior of the system, groups of similar Elementary Processes are lumped together. In the following example, it is to be investigated how the percentage of I/O Elementary Processes in the Mix influences the performance of the system, where the percentage of I/O Elementary Processes is defined as the sum of the weights of all I/O Elementary Processes in a Mix. Five different Mixes were constructed consisting of 10, 30, 50, 70, and 90 percent I/O Elementary Processes, respectively. The resulting Throughput Rates and Gain Factors are depicted in Figures 4 and 5 respectively. The parameter r denotes the number of jobs simultaneously active in the system (degree of multiprogramming).

Figure 5 shows that the Gain Factor increases with the percentage of I/O, or in other words, Mixes with higher I/O percentages (I/O bound Mix) overlap their Elementary Processes more than those with lower I/O percentages (compute bound Mix). The slow increase between 30 and 90 percent I/O

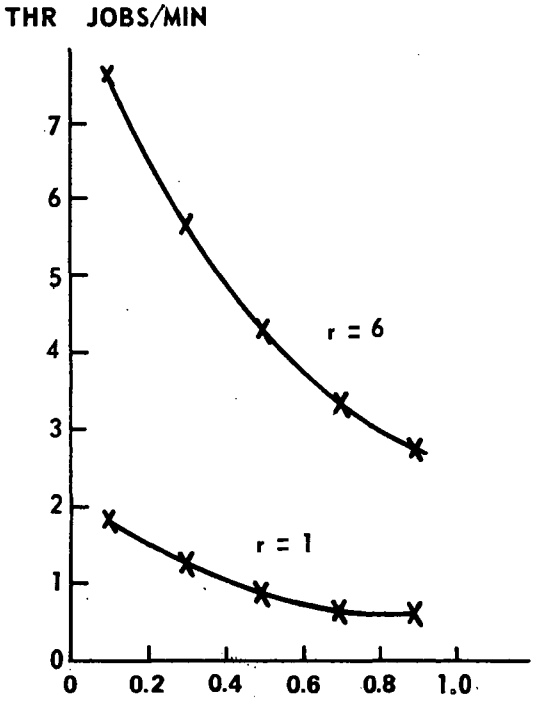


Figure 4. Percentage I/O Throughput Rates.

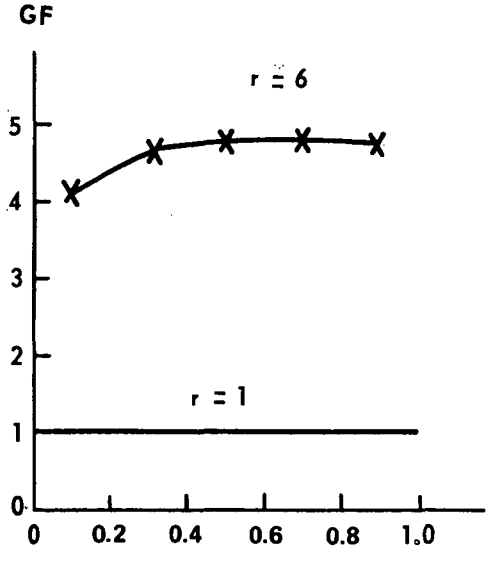


Figure 5. Percentage I/O Gain Factors.

is due to the fact that the Types in each of the Mixes used the same I/O channels. Despite this relatively better utilization of the system processors by I/O bound Mixes, a much greater number of compute bound jobs leaves the system per unit of time.

This paradox behavior can be explained if we recall that those Elementary Processes which use inexpensive Virtual Processors will be executed for relatively longer periods of time. The Throughput Rates in Figure 4, therefore, include both the effects of overlapping Elementary Processes and the cost of the processors used. From Figure 5 it can be concluded that compute bound Mixes utilize the more expensive processors of the system much better than I/O bound Mixes. The cost structure of the system is of greater influence on the Throughput per Dollar than the degree of overlapped processor usage expressed by the Gain Factor. In other words, if the system runs an I/O bound Mix, it wastes the more expensive processors available in the system. Consequently, a time rental service would have to apply a factor of 1.7 to the price of a 50 percent I/O Mix in order to draw the same income from this Mix as it does from compute bound jobs, if it uses this architecture.

If the configuration and the architecture of a machine cannot be changed but the workload can be varied, the performance curves could indicate an optimum workload for a machine. Since Figure 4 does not show a true optimum, the most cost effective workload on this machine consists of an extremely compute bound batch of programs.

Second, to investigate the influence of specific I/O Elementary Processes on Throughput Rate and Gain Factor, three Mix Types were constructed, each of which contains 50 percent I/O Elementary Processes. The first one uses exclusively slow drum channels (Fastrand), the total I/O percentage of the second one is generated by equal percentages of I/O activities on slow and fast drum channels (Fastrand FH-1782), and the third one uses only fast drum channels (FH-1782). The results are shown in Figures 6 and 7 where it can be seen that the workload becomes more cost effective and utilizes the processors to a higher extent when the I/O activities are predominantly handled by high speed channels.

The third class of problems is concerned with the selection of a computer system or of a system configuration best suited for handling a given workload. In this case one computer system will be considered as the reference system and Throughput Rates and Gain Factors of all other machines are expressed relative to the performance of this reference system. A family of curves similar to those presented in Figures 4 and 5 will be obtained.

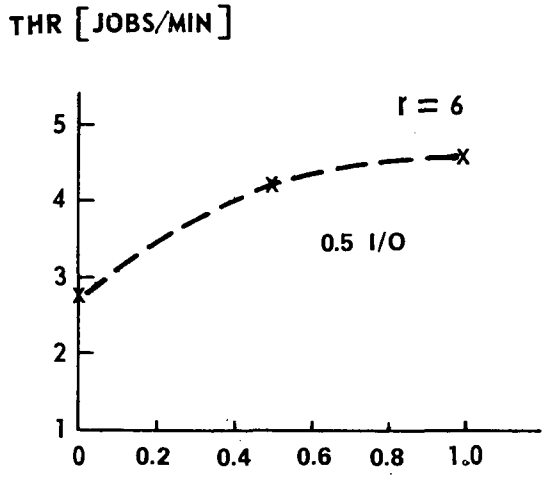


Figure 6. Weight of Fast Drum Process relative to weight of total I/O, THR.

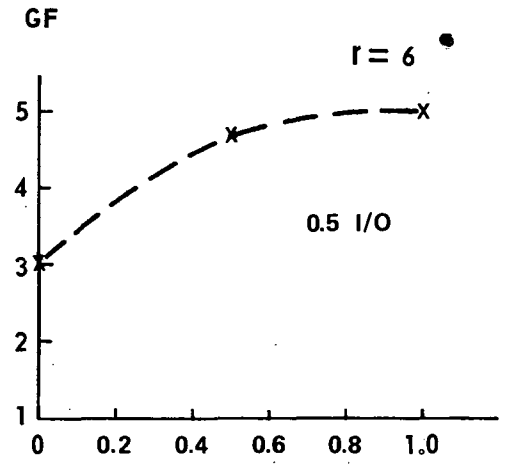


Figure 7. Weight of Fast Drum Process relative to weight of total I/O, GF.

Finally, the performance of a system will depend not only on its architecture or the general type of workload applied but also on the structure of its programs. The Types of a Mix indicate the structure of the programs to be simulated in their transition tables, especially through the degree of hashing between Elementary Processes and also through their sequencing. In order to study the influence of hashing, one of the previously used Mixes (50 percent I/O) was repeated with different degrees of hashing of its Elementary Processes without changing the weight of its constituent Elementary Processes (Figs. 8 and 9). The results show decreasing Throughput Rates and Gain Factors when the degree of hashing is increased. This effect is mainly due to an increased system overhead.

CONCLUSIONS

This report presents definitions for measuring the performance of a complex computer system and a method for constructing workloads with desired properties. The workloads so constructed show in a very translucent way the degree of subsystem involvement, the structure of the program, and the constraints imposed by storage requirements for programs or batches of programs. The proposed workload is easily transferable between different machines. It allows the comparison of the performance of different workloads

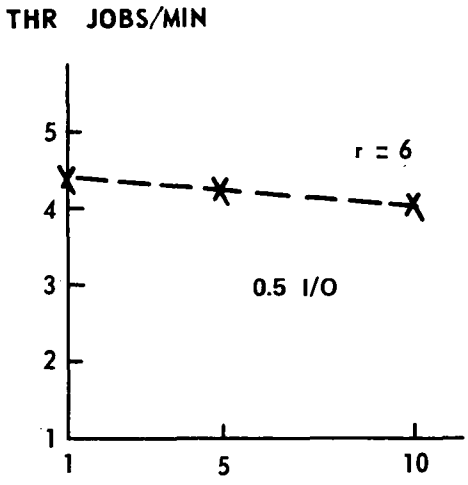


Figure 8. Throughput Rate with degree of hashing.

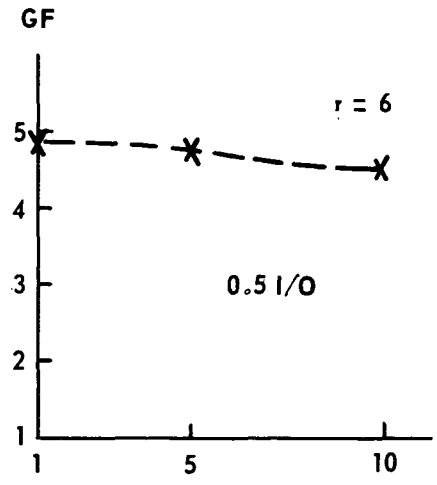


Figure 9. Gain Factor with degree of hashing.

on the same machine and reveals the cost effectiveness of an architecture relative to the workload. Because of the extreme visibility of its structure and its contributing Elementary Processes, such a workload is also useful in searching for bottlenecks in a system. Since it is easy to construct a workload of a prescribed property, this method may be used for experimental verification of theoretical studies on system performance. The generalized parameters expressing the weight or structure of a program could allow the description of a workload sufficient to compare the results of experimental studies on different systems by different authors. The parameters found to be essential for the performance of a given architecture can be used as the basis for adaptively scheduling the workload inputs to a large scale system for performance optimization.

George C. Marshall Space Flight Center
National Aeronautics and Space Administration
Marshall Space Flight Center, Alabama, February 11, 1972

REFERENCES

1. Auerbach Corporation: Auerbach Standard EDP Reports. Auerbach Information, Inc., Philadelphia, Pa.

2. Buchholz, W.: A Synthetic Job for Measuring System Performance. IBM Syst. J., no. 4, 1969, pp. 309-318.

3. Chang, W.: Queues with Feedback for Time-Sharing Computer System Analysis. Operations Research, 1968, pp. 613-627.

4. Gaver, D. P.: Probability Models for Multiprogramming Computer Systems. Journal of the ACM, 1967, pp. 423-438.

5. Katz, J. H.: An Experimental Model of System/360. Communications of the ACM, 1967, pp. 694-702.

6. Kleinrock, L.: Time-Shared Systems: A Theoretical Treatment. Journal of the ACM, 1967, pp. 242-261.

7. Kuemmerle, K.: Simulation of the Performance of Computer Systems. Elktron. Rechenanl., Vol. 12, 1970, pp. 324-328.

8. Nielsen, N. R.: The Simulation of Time-Sharing Systems. Communications of the ACM, 1967, pp. 397-412.

NASA-Langley, 1972 —— 8

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
WASHINGTON, D.C. 20546

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE $300

SPECIAL FOURTH-CLASS RATE BOOK

POSTAGE AND FEES PAID
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
451

U.S.MAIL

POSTMASTER: If Undeliverable (Section 158 Postal Manual) Do Not Return

"The aeronautical and space activities of the United States shall be conducted so as to contribute . . . to the expansion of human knowledge of phenomena in the atmosphere and space. The Administration shall provide for the widest practicable and appropriate dissemination of information concerning its activities and the results thereof."

—NATIONAL AERONAUTICS AND SPACE ACT OF 1958

NASA SCIENTIFIC AND TECHNICAL PUBLICATIONS

TECHNICAL REPORTS: Scientific and technical information considered important, complete, and a lasting contribution to existing knowledge.

TECHNICAL NOTES: Information less broad in scope but nevertheless of importance as a contribution to existing knowledge.

TECHNICAL MEMORANDUMS: Information receiving limited distribution because of preliminary data, security classification, or other reasons. Also includes conference proceedings with either limited or unlimited distribution.

CONTRACTOR REPORTS: Scientific and technical information generated under a NASA contract or grant and considered an important contribution to existing knowledge.

TECHNICAL TRANSLATIONS: Information published in a foreign language considered to merit NASA distribution in English.

SPECIAL PUBLICATIONS: Information derived from or of value to NASA activities. Publications include final reports of major projects, monographs, data compilations, handbooks, sourcebooks, and special bibliographies.

TECHNOLOGY UTILIZATION PUBLICATIONS: Information on technology used by NASA that may be of particular interest in commercial and other non-aerospace applications. Publications include Tech Briefs, Technology Utilization Reports and Technology Surveys.

Details on the availability of these publications may be obtained from:

SCIENTIFIC AND TECHNICAL INFORMATION OFFICE

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Washington, D.C. 20546