On the Construction of a Representative Synthetic Workload

K. Sreenivasan and A.J. Kleinman
The MITRE Corporation

A general method of constructing a drive workload representative of a real workload is described. The real workload is characterized by its demands on the various system resources. These characteristics of the real workload are obtained from the system accounting data. The characteristics of the drive workload are determined by matching the joint probability density of the real workload with that of the drive workload. The drive workload is realized by using a synthetic program in which the characteristics can be varied by varying the appropriate parameters. Calibration experiments are conducted to determine expressions relating the synthetic program parameters with the workload characteristics. The general method is applied to the case of two variables, cpu seconds and number of I/O activities; and a synthetic workload with 88 jobs is constructed to represent a month's workload consisting of about 6000 jobs.

Key Words and Phrases: benchmarks, calibration, drive workload, input to simulation, probability distribution, representative workload, synthetic workload, workload characteristics

CR Categories: 5.5, 5.9, 8.1

This work was supported by Air Force Contract F19628-73-C-0001. Authors' address: Management and Computer Systems, The MITRE Corporation, Bedford, MA 01730.

Introduction

The evaluation of computer systems is usually conducted for the purposes of improving the present performance, predicting the effects of changes in either the existing system or the workload, or sizing and selecting a new computer system. The evaluation may use analytical modeling, simulation, or experiments to be performed on the existing system. In all these cases, there is a need for a drive workload that imitates the actual workload with reasonable fidelity, but in an abbreviated form, e.g. less time or fewer jobs. A method of constructing such a drive workload is described.

Workload is defined as the collection of all individual jobs and data that are processed by the computer system during a specified period of time. The term "workload characteristics" refers to the demands placed on the various system resources. The magnitudes of these demands are determined by the nature of the individual jobs, and can be conveniently used to quantify the workload. For example, some of the characteristics are the processor time (cpu seconds) needed, the number of I/O activities initiated, the core requirements, and the usage of unit record devices. The observed performance of a computer system is directly dependent upon the characteristics of the drive workload which must be representative of the actual workload for valid conclusions to be drawn from its use.

Many methods for constructing drive workloads are described in the literature. Lucas [1] discusses synthetic programs in the context of performance evaluation and monitoring. Joslin [2] describes a technique for selecting representative benchmarks by classifying the workload according to the type of job, and then selecting a combination of jobs to represent the characteristics of each class. Shope et al. [3] constructed a drive workload consisting of jobs selected from the actual workload by statistical sampling, and then adjusted this collection until it was considered representative. Wood and Forman [4] composed a synthetic workload using Shope's technique, and substituting Buchholz's synthetic program [5] for each job in the mix. The parameters of the synthetic program seem to have been determined by trial and error. Recent proposals for creating an industry-wide library of standardized benchmarks [6] and a library of synthetic modules [7] may reduce the number of man hours required to code and debug the test programs, but there still remains the need for a method of constructing a representative workload from such a collection. Ferrari [8] stresses the importance of workload characterization and points out the need for

methods of constructing drive workloads that are representative of real workloads. Kernighan and Hamilton [9] describe the design of and experience with an automated benchmark-generation facility for system testing and performance evaluation. Morgan and Campbell [10] present a method of constructing synthetic jobs to simulate real jobs on a one-to-one basis. The methods described in these papers for selecting jobs to constitute a drive workload seem somewhat arbitrary and have not been shown to generate a representative workload.

In our work, the real workload is characterized by the magnitude of its demands on the system resources. These characteristics are obtained from the system accounting data. The characteristics of the drive workload are determined by matching the joint probability density of the real workload with that of the drive workload. The cpu time and the number of I/O activities are the two characteristics selected to quantify the workload. Each job in the drive workload is realized by using a synthetic program. Calibration experiments are conducted to determine expressions relating the synthetic program parameters with the workload characteristics.

Characteristics of the Workload

One way to classify jobs processed by a computer system is according to the type of processing, e.g. compilation, assembly, sort-merge, matrix inversion, and file update. The distribution of jobs among these groups describes the nature of the workload. This description does not depend on the type of the computer system, and hence, can be referred to as a system independent description. In this method of description, a typical workload is represented by the set of numbers $n_1, n_2, \ldots, n_k$, where the subscripts 1, 2, 3, ..., k refer to the job classes chosen and the numbers $n_1, n_2, \ldots, n_k$ denote the number of jobs in each class.

Using this description, a representative workload can be constructed by scaling down the number of jobs in each class. This method of representation is seldom feasible because it is very difficult to obtain information on the type of each job. Even if such information was available, the use of actual jobs in a drive workload has the following disadvantages: (a) the drive workload is not flexible, as it is constructed from jobs with fixed characteristics; (b) it is difficult to make all the jobs operational because their interfaces with a new operating system, for example, are no longer compatible, and may have to be modified; (c) duplication of large amounts of data on auxiliary storage is expensive; and (d) security or privacy considerations may prevent the use of some jobs.

Alternatively, a computer system can be considered a collection of resources upon which the workload places a demand. The resources common to many computer installations and the corresponding demands are: (a) processor (cpu seconds); (b) I/O channels and devices (number of I/O activities initiated); (c) core memory (amount of core allocated); and (d) unit record devices (number of cards read, cards punched, and lines printed).

The demands on these resources can be considered the characteristic variables of the real workload processed by the computer system. A job, such as compilation, matrix inversion, or sort-merge can then be described by a set of these variables whose magnitude will vary from one type of job to another, and from one computer system to another; for example the cpu seconds necessary are a function of the speed of the processor. In this method of characterizing the workload, it is assumed that since the system does not recognize the type of job, two jobs with the same values of these characteristic variables are treated identically by the operating system.

Many computer installations maintain, for charging purposes, a system accounting package that collects information about the use of various resources. This is a ready source of data from which many of the workload characteristics can be derived. To determine the performance characteristics not available from the accounting data, hardware and software monitors must be used. The source of accounting data for this paper was the System Management Facility (SMF) which maintains the accounting data for the IBM 370/155 at MITRE (Bedford), and reports the cpu seconds, number of EXCP (Execute Channel Program), amount of core used, etc., for each job processed.

The number of EXCP is approximately equal to the number of blocks of data transferred to and from the auxiliary storage. For random access devices, e.g. disks, this approximation leads to negligible errors because the difference between the number of EXCP and the number of blocks transferred is equal to the end-of-file signals given, which are equal to the files closed during execution. For sequential access devices, e.g. tapes, not all the EXCP lead to data transfers; some result in positioning, rewinding, and label checking. A large number of jobs in the real workload considered in this paper use random access devices and, hence, it is reasonable to equate the number of EXCP to the number of blocks of data transferred to and from the auxiliary storage.

Representative Modeling of the Workload

The characteristics of a representative model workload must resemble those of the actual workload with reasonable fidelity while satisfying several practical constraints. The characteristics used depend on the purpose of the evaluation study, the system being studied, and the availability and cost of various performance monitors. "Representativeness" may be viewed as a measure of the closeness of fit between the model, i.e. the drive workload, and the actual workload, thus

creating a need for a distance function. This distance function is influenced by the number of characteristics chosen in the representation. The constraints on the model may be in the form of limits on the total elapsed time, cpu time, or the number of jobs in the drive workload. Thus, there is a need for a compromise between representativeness and practicality.

Previous approaches to the solution of this problem are based upon statistical sampling, functional grouping, and cluster analysis. In the statistical sampling method [3], jobs are selected at random from the actual workload with the aim of composing a one-hour benchmark that "represented" the real workload for a performance improvement effort. The process of selecting the jobs in this method seems to rely on trial and error. Joslin [2] composes the representative application benchmarks by dividing the workload into several classes based on job functions—e.g. business, mathematics, and engineering—and then picking a job that has the characteristics of the weighted average for that group. An improved method of partitioning the set of jobs into classes would be to use a clustering algorithm using actual performance data [11]. Presently available clustering algorithms, however have several major limitations; for example, computation time increases excessively as the number of dimensions and cluster groups increases.

The method used in this study was to choose a representative number of jobs by matching the joint probability density function of the drive workload with that of the real workload, subject to the constraint that the total cpu time should be about T seconds.

In general, let $X_1, X_2, \ldots, X_S$ be the S variables used to characterize the workload. Each job in the workload can be described by a set of these S parameters. A single job may be regarded as a point in the S-dimensional space whose coordinate axes represent each demand. An ensemble of jobs, i.e. the workload, is described by a set of points in this coordinate system. An alternate method of description consists in dividing the region into a finite number of discrete cells and computing for each cell its percentage of the total number of jobs—the joint probability density of the workload. Let each axis be divided into L parts, thus resulting in L^S discrete cells. Let $N_{ijk\ldots r}$ be the number of jobs in the $(i, j, k, \ldots, r)$-th cell with coordinates $X_1 = x_1^{(i)}$, $X_2 = x_2^{(j)}, \ldots, X_S^{(r)} = x_S^{(r)}$ for $i, j, k, \ldots, r = 1, 2, 3, \ldots, L$. The magnitude of the variables corresponding to each interval is taken to be its value at the mid-point of the interval. The probability of finding a job in the $(i, j, k, \ldots, r)$-th cell is equal to the joint probability density, and is given by

$$P_{ijk\ldots r} = N_{ijk\ldots r}/N_{\text{tot}}, \qquad i, j, k, \ldots, r = 1, 2, 3, \ldots, L, \tag{1}$$

where N_{tot} is the total number of jobs in the real workload.

Using the values of $P_{ijk\ldots r}$ from eq. (1), the values of $N'_{ijk\ldots r}$ (primed quantities refer to the drive workload) for the drive workload are obtained from:

$$N'_{ijk\ldots r} = P_{ijk\ldots r}N'_{\text{tot}}, \qquad i, j, k, \ldots, r = 1, 2, 3, \ldots, L. \tag{2}$$

Equation (2) can be physically interpreted as matching the joint probability density of the drive workload with that of the real workload. The constraint on the total cpu time (X_1-axis) can be expressed as

$$\sum_{i,j,k,\cdots,r} x_i^{(i)} N'_{ijk\ldots r} \simeq T, \tag{3}$$

where $x_i^{(i)}$ is the X_1-value for the $(i, j, k, \ldots, r)$-th cell. The summation indicated on the left side of eq. (3) is merely the sum of the products of the number of jobs in each of the L^S cells and its corresponding value of X_1.

Using an initial guess for N'_{tot}, the values of $N'_{ijk\ldots r}$ from eq. (2) are substituted in eq. (3) to check the constraint. Depending on this comparison, the initial guess for N'_{tot} is changed, and the procedure repeated until the constraint is satisfied.

A drive workload constructed by matching the joint probability distribution consists of the most frequently occurring jobs. In many real situations, there may be jobs that occur infrequently but place heavy demands on the system resources. These infrequent jobs can be accounted for by using a weighted joint probability density, $q_{ijk\ldots r}$. If the infrequent jobs use resource X_1 heavily, the weighted probability density function is defined as

$$q_{ijk\ldots r} = x_1^{(i)} P_{ijk\ldots r}/E(X_1), \tag{4}$$

where $E(X_1)$ is the average value of X_1 for all the jobs in the real workload and is given by

$$E(X_1) = \sum_{i,j,k,\cdots,r} x_1^{(i)} P_{ijk\ldots r}. \tag{5}$$

The characteristics of the drive workload (primed quantities) are determined by matching the weighted probability density, and are given by

$$N'_{ijk\ldots r} = q_{ijk\ldots r}N'_{\text{tot}}, \qquad i, j, k, \ldots, r = 1, 2, 3, \ldots, L. \tag{6}$$

The use of the weighted probability density is influenced by the nature of the distribution of the selected variables in the real workload.

Two variables, cpu seconds (X_1) and the number of EXCP (X_2), were selected to characterize the workload of the MITRE IBM 370/155. The amount of core utilized is another important characteristic, but was not selected for representation because it was observed that over 80 percent of the jobs in the real workload requested a 128-kilo byte partition. At MITRE such jobs run with the highest priority and most users are thus persuaded to request this minimum partition. The core requirement for the MITRE workload can then be conveniently represented by a uniform demand of 128 kilo bytes.

The characteristics of the drive workload were obtained by matching the unweighted joint probability density of cpu seconds (X_1) and the number of EXCP

(X_2). A total cpu time of 1800 seconds was imposed as the constraint. Equations (1), (2), and (3) can be easily applied to this by noting that $S = 2$ and $T = 1800$. The resulting equations are:

$$P_{ij} = \frac{N_{ij}}{N_{\text{tot}}}, \qquad i, j, = 1, 2, 3, \ldots, L, \tag{7}$$

$$N'_{ij} = P_{ij}N'_{\text{tot}}, \qquad i, j, = 1, 2, 3, \ldots, L, \tag{8}$$

$$\sum_{i,j} x_1^{(i)} N_{ij} \simeq 1800. \tag{9}$$

The SMF summary data used in this study contains the cpu seconds (X_1) and the number of EXCP (X_2) for 6126 jobs. Figure 1 shows this bi-variate distribution with Log X_1 and Log X_2 as the coordinate axes. The numbers in the figure indicate the number of jobs in each cell. The value of N'_{tot} was found to be 88, and the joint probability distribution for the drive workload is shown in Figure 2.

Synthetic Program

The synthetic workload is a collection of individual synthetic programs just as a real workload is a collection of individual jobs. The synthetic program is a contrivance used to transform the drive workload characteristics into a set of parameters. The choice of the synthetic program is dictated by the nature of the application, the number of variables to be represented, and the ease of calibration and inversion.

A FORTRAN synthetic program simulating a file updating process was used in this study. The program was first written in PL/I by Buchholz [5] and later converted to FORTRAN [12].

The synthetic program simulates a general file update process using four data sets. Initially two data sets, one master and one detail, are created. They are the input to the file update process. During the update process, the master and detail records are sequentially read. For each detail record, one or more master records are searched for a match. When the matching master record is found, the compute kernel is executed. Next, the updated detail and master records are written out to two output data sets.

The parameters of the synthetic program, P_1 through P_6, are defined as follows:

(a) P_1 is the number of master records, NMAS.
(b) P_2 is the number of detail records, NDET.
(c) P_3 is the number of times the compute kernel is executed per match, NCPURP.
(d) P_4 is the number of times the file update process is repeated, NPAS.
(e) P_5 is the blocksize of the I/O buffers.
(f) P_6 is the size of records.

The magnitude of these parameters determines the load on the system, i.e. the cpu seconds used and the number of EXCP issued by the synthetic program. These parameters can be manipulated so as to obtain either a cpu-bound job, or an I/O-bound job, or a job anywhere in between these two extreme cases.

P_1, P_2, P_3, P_4, P_5, and P_6 are the independent parameters for the synthetic program; the cpu seconds (X_1) and number of EXCP (X_2) are the two dependent parameters. The functional dependence of X_1 and X_2 on the parameters can be expressed as

$$X_1 = K_1 + K_2(P_4) + K_3(P_1 + P_2) + K_4(P_4)(P_1 + P_2) + K_5(P_2)(P_3)(P_4), \tag{10}$$

$$X_2 = 2P_4 + (2P_4 + 1)\{[P_1P_6/P_5] + [P_2P_6/P_5]\}. \tag{11}$$

The form of eq. (10) was determined by a close examination of the synthetic program. X_1 is proportional to the number of instructions executed, which in turn can be divided into the following groups: P_4 is the number of times the initial code is executed; (P_1+P_2) corresponds to the number of records initially created; $P_4(P_1+P_2)$ corresponds to the total number of reads and writes; (P_2) (P_3) (P_4) refer to the number of kernel executions. A linear combination of these groups was used in this paper and found sufficiently accurate. The form of eq. (11) was obtained as follows: The number of records per block is $[P_6/P_5]$,[1] and the number of blocks read (or written) to the master and detail files is equal to $[P_1P_6/P_5]$ and $[P_2P_6/P_5]$, respectively; these blocks are written out during the initial creation and every time the files are updated the number blocks read (or written) is given by $2\{[P_1P_6/P_5] + [P_2P_6/P_5]\}$; the number of EXCP from closing the two files is equal to 2; the total number of EXCP (X_2) for P_4 file updates is given by P_4 times the sum of the above terms.

Calibration of the Synthetic Program

Calibration experiments were performed to determine the constants in eq. (10) and check the validity of eqs. (10) and (11). The values of the independent parameters are specified for each run, and the resulting values of the cpu seconds and the number of EXCP are obtained from the accounting data. Parameters P_1, P_2, P_3, and P_4 are specified in a data card read by the program; P_5 and P_6 are specified in the JCL (Job Control Language) cards. A set of 62 experiments was conducted in which the independent parameters were varied in the following range:

NMAS(P_1)	0 to 5000	NPAS(P_4) 0 to 10,
NDET(P_2)	0 to 1000	Blocksize (P_5) 440 to 6582 bytes,
NCPURP(P_3)	0 to 10,000	Record size (P_6) 206 bytes.

Using these experimental results, the constants K_1, K_2, K_3, K_4, and K_5 were determined by a linear regression procedure. The values of these constants in seconds, for the IBM 370/155, are: $K_1 = 0.28$; $K_2 = 0.5$; $K_3 = 0.00092$; $K_4 = 0.00138$; and $K_5 = 0.00024$.

[1] [Z] is the ceiling function of Z.

Fig. 1. Joint probability distribution of cpu seconds and number of EXCP for the real workload.

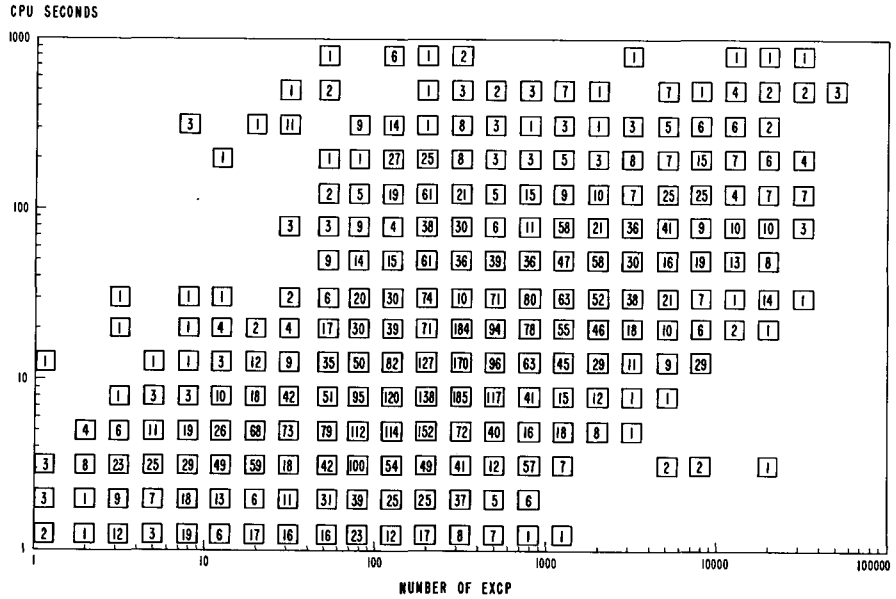

Fig. 2. Joint probability distribution for the drive workload.

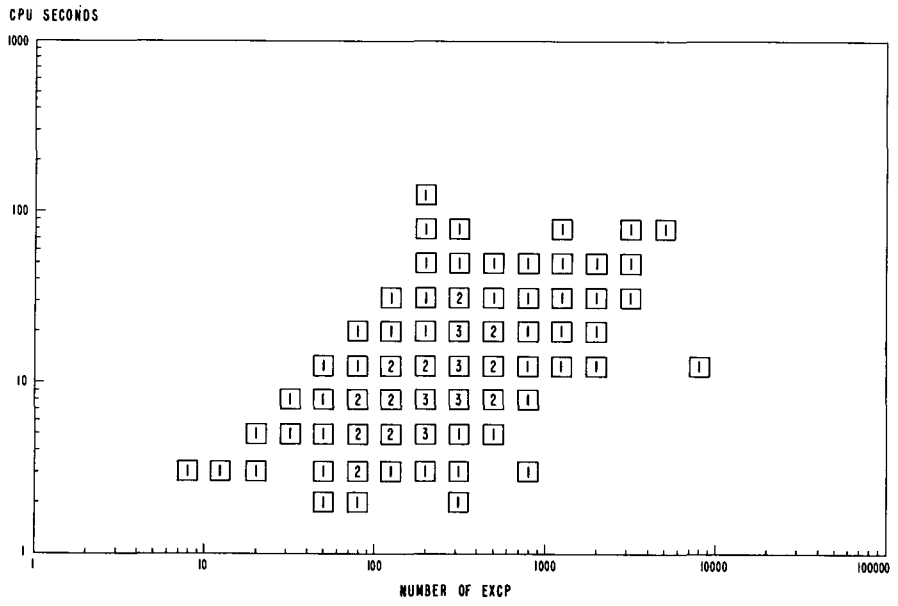

Table I.

	Required Drive Workload Characteristics		Parameter Settings for the Synthetic Workload					Synthetic Workload Characteristics	
Number	X_1 CPU Seconds	X_2 Number of EXCP	P_1(NMAS)	P_2(NDET)	P_3(NCPURP)	P_4(NPAS)	P_5 (Blocksize)	X_1 CPU Seconds	X_2 Number of EXCP
1	2.00	51	102	51	73	1	2088	2.00	53
2	2.00	80	170	85	32	1	2088	1.99	80
3	2.00	317	146	73	42	1	240	1.99	317
4	3.16	13	64	8	1168	1	6582	3.16	14
5	3.16	20	32	16	598	1	2088	3.16	20
6	3.16	51	320	160	34	1	6582	3.16	53
7	3.16	80	208	52	145	1	2088	3.16	83
8	3.16	126	63	21	439	1	440	3.16	122
9	3.16	200	92	46	189	1	440	3.15	200
10	3.16	317	148	74	106	1	440	3.14	317
11	3.16	795	448	112	41	1	440	3.14	797
12	5.01	20	39	13	1327	1	2088	5.01	20
13	5.01	32	250	50	297	1	6582	5.00	32
14	5.01	51	102	51	319	1	2088	5.01	53
15	5.01	80	170	85	180	1	2088	5.01	80
16	5.01	126	270	135	102	1	2088	4.99	125
17	5.01	200	102	34	483	1	440	5.00	200
18	5.01	317	146	73	214	1	440	5.00	317
19	5.01	502	234	117	122	1	440	4.98	503
20	7.94	32	60	30	969	1	2088	7.93	29
21	7.94	51	124	31	918	1	2088	7.94	53
22	7.94	80	693	99	225	1	6582	7.92	80
23	7.94	126	850	425	41	1	6582	7.86	125
24	7.94	200	12	6	193	10	440	7.93	209
25	7.94	317	700	350	56	1	2088	7.87	317
26	7.94	502	30	15	66	10	440	7.93	503
27	7.94	795	1766	883	5	1	2088	7.90	794
28	12.59	51	120	40	1195	1	2088	12.59	50
29	12.59	80	170	85	551	1	2088	12.58	80
30	12.59	126	854	427	86	1	6582	12.51	125
31	12.59	200	160	80	21	10	6582	12.45	209
32	12.59	317	348	87	6	10	6582	12.55	314
33	12.59	502	264	88	522	1	440	12.58	503
34	12.59	795	1766	883	27	1	2088	12.56	794
35	12.59	1259	82	41	59	10	440	12.51	1259
36	19.95	126	270	135	563	1	2088	19.92	125
37	19.95	200	440	220	334	1	2088	19.90	200
38	19.95	317	2214	1107	43	1	6582	19.81	317
39	19.95	502	3486	1743	17	1	6582	19.89	503
40	19.95	795	1766	883	61	1	2088	19.77	794
41	19.95	1259	590	295	242	1	440	19.92	1259
42	19.95	1996	624	312	1	10	2088	19.42	2015
43	19.95	200	434	217	563	1	2088	31.57	200
44	31.62	317	99	33	312	10	2088	31.54	314
45	31.62	502	1390	278	405	1	2088	31.61	503
46	31.62	795	5566	2783	17	1	6582	31.31	794
47	31.62	1259	82	41	253	10	440	31.60	1259
48	31.62	1996	132	66	150	10	440	31.56	2015
49	31.62	3163	1671	557	192	1	440	31.54	3164
50	50.12	200	1380	690	269	1	6582	50.06	200
51	50.12	317	840	210	931	1	2088	50.09	317
52	50.12	502	3482	1741	89	1	6582	49.95	503
53	50.12	795	1766	883	204	1	2088	50.07	794
54	50.12	1259	435	145	105	10	2088	49.97	1259
55	50.12	1996	15756	5252	0	1	6582	49.07	1994
56	79.43	200	440	220	1461	1	2088	79.41	200
57	79.43	1259	492	82	335	10	2088	79.27	1259
58	79.43	3163	7046	3523	64	1	2088	79.17	3161
59	79.43	5012	1586	793	20	10	2088	77.97	4997
60	125.89	200	1390	695	721	1	6582	125.81	200

The constant K_5 which corresponds to the execution time of the compute kernel was verified independently by counting the number of each type of instruction executed in the object code, and using the timing data in the IBM manual [13]. The calculated value of K_5 was 0.00025 which agrees well with the experimentally determined value (0.00024). This good agreement shows that the form of eq. (10) and the experimental method of determining the constants are reasonably accurate for our purposes. The value of X_2, predicted by eq. (11), agrees with the values recorded by SMF.

Construction of Synthetic Workload

The representative synthetic workload is a collection of synthetic programs which realize each of the N'_{tot} jobs in the representative model described earlier. In this section, we determine the required synthetic program parameter settings (P_1, P_2, P_3, P_4, P_5, P_6) which correspond to the values of the workload characteristics (X_1, X_2) for each job in the synthetic workload. In formal terms, given the values of X_1, X_2 and the equations describing the behavior of the synthetic program,

$$X_1 = 0.28 + 0.5\,P_4 + 0.00092(P_1 + P_2) + 0.00138\,P_4(P_1 + P_2) + 0.00024\,P_2P_3P_4, \quad (12)$$

$$X_2 = 2P_4 + (2P_4 + 1)\{[P_1P_6/P_5] + [P_2P_6/P_5]\}, \quad (13)$$

the problem is to find P_1, P_2, P_3, P_4, P_5, and P_6 by solving the above system of equations.

Since there are only two equations relating six variables, more than one set of parameters P_1, P_2, . . ., P_6 may be used to generate a pair of X_1 and X_2. The following procedure was used to solve for P_1, P_2, . . ., P_6.

Integral values for P_1, P_2, P_5, and P_6 were chosen, and using the given value of X_2, eq. (13) was solved for P_4. A nonintegral value of P_4 was rounded to the nearest integer, and the value of X_2 was calculated using eq. (13). If the calculated and the given values of X_2 were not equal, P_1 and P_2 were slightly altered to make them equal. It should be noted that P_1 and P_2 can be varied over a small range (the range depends on P_5 and P_6) without altering the ceiling functions on the right-hand side of eq. (13). If P_4 was found to be negative, a new set of integral values of P_1, P_2, P_5, and P_6 was chosen, and the above procedure was repeated. Using these values of P_1, P_2, and P_4 and the given value of X_1, eq. (12) was solved for P_3. A nonintegral value of P_3 was rounded to the nearest integer, and the value of X_1 was calculated using eq. (12). If the calculated and the given values of X_1 were not equal, P_1 and P_2 were slightly altered to make them equal. If P_3 was found to be negative, a fresh start was made with another choice of the parameters.

Based on the values of X_1 and X_2 for each group of N'_{jk} jobs, the values of P_1, P_2, P_3, P_4, P_5, and P_6 are obtained. Table I lists the values of the parameters for the 88 synthetic jobs that constitute the synthetic workload for the case under study, namely, a month's workload for the IBM 370/155 at MITRE (Bedford).

Conclusions

The paper presents methods of quantifying the workload processed by a computer system, in terms of the workload characteristics—the magnitudes of the demands on the various system resources. These characteristics are, typically, the cpu seconds, number of I/O activities initiated, amount of core utilized, and the usage of unit record devices; they are determined from the system accounting data.

A drive workload is constructed to represent the real workload by matching the joint probability distribution of the real workload with that of the drive workload. Although cpu seconds and the number of I/O activities are the two parameters chosen for this representation, the method developed is general in scope and can be applied to more than two variables.

The drive workload was realized by using a synthetic program. The synthetic program has several parameters, which are manipulated to achieve the desired workload characteristics.

Since the synthetic workload is only a model, deliberately constrained in cpu time used, it cannot be completely representative of all the features of the real workload. In this study, we used a set of jobs with different cpu-I/O characteristics, but all created from the same synthetic program. Hence, all the jobs invoke the same system tasks, namely, opening and closing data sets, reading and writing to data sets, etc. The synthetic workload may not exercise all the available features of the operating system. These possible disadvantages can be avoided by several synthetic programs exercising different features of the operating system and using different compute kernels.

In systems where paging plays an important role, the synthetic program used in this paper is inadequate, and new synthetic programs must be designed.

The above discussion suggests that there is a need for a synthetic job library that contains several types of synthetic programs, each representing one class of problems. Users will then be able to select the appropriate jobs from this library and, by adopting the method of representation described in this paper, construct their own synthetic workload.

The synthetic workload has several applications in the general area of computer performance. It can be used as the input to simulation studies, analytical models, or experiments with existing systems. The relative performance of dissimilar systems can be evaluated. Effects of external scheduling, multiprogramming, and allocation of the data sets on different devices and channels can be studied. Such experiments help in anlayzing the effects of device and channel contention. The effect of changes in or additions to the existing configuration can be studied.

Experiments are being performed along these lines, and the results will be presented in a forthcoming paper. The success of this method of approach can probably be assessed by the following factors: the ease with which the accounting data are reduced to determine the drive workload characteristics; the flexibility incorporated into the drive workload; and the ability to translate the experimental results obtained from the drive workload into real-life situations.

Acknowledgments. The authors sincerely thank J.E. Esposito for the programming help, J. Mitchell for his constructive suggestions, and O.R. Kinney and L.M. Thomas for their encouragement during the several phases of the investigation—all of The MITRE Corporation. The authors are grateful to the referees for their critical remarks.

Received April 1973; revised July 1973

References

1. Lucas, H.C., Jr. Performance evaluation and monitoring. *Computing Surveys, 3*, 3 (Sept. 1971), 79–91.
2. Joslin, E.O. Application benchmarks: The key to meaningful computer evaluations. Proc. 20th ACM Nat. Conf., 1965, pp. 27–37.
3. Shope, W.L., Kashmark, K.L., Inghram, J.W., and Decker, W.F. System performance study. Proc. SHARE XXXIV, Vol. 1, Colorado, Mar. 1970, pp. 439–530.
4. Wood, D.C., and Forman, E.H. Throughout measurement using a synthetic job stream. Proc. AFIPS 1971 FJCC, Vol. 39, AFIPS Press, Montvale, N.J., pp. 51–56.
5. Buchholz, W. A synthetic job for measuring system performance. *IBM Syst. J. 8*, 4 (1969), 309–318.
6. Lucas, H.C., Jr. Synthetic program specifications for performance evaluation. Proc. ACM Nat. Conf., Boston, 1972, pp. 1041–1058.
7. Gamse, R.N. Approach plan for a standard benchmark library for use in computer system-selection. MITRE Tech. Rep., MTR-2226, Sept. 1971.
8. Ferrari, D. Workload characterization and selection in computer performance measurement. *Computer* (July/Aug. 1972), 18–24.
9. Kernighan, B.W., and Hamilton, P.A. Synthetically generated performance test loads for operating systems. 1st Ann. SIGME Symp. on Measurement and Evaluation, 1973, pp. 121–126.
10. Campbell, J.A., and Morgan, D.E., An answer to a user's plea? 1st Ann. SIGME Symp. on Measurement and evaluation, 1973, pp. 112–120.
11. Hunt, E., Hiehr, G., and Garnatz, D. Who are users?—An analysis of computer use in a university computer center. Proc. AFIPS 1971 SJCC, Vol. 38, AFIPS Press, Montvale, N.J., pp. 231–238.
12. FORTRAN version of Buchholz's synthetic program. Transl. by J. Maranzano and modified by T.E. Bell. Proc. 3rd Meeting of the Computer Performance Evaluation Users Committee, July 1971, Washington, D.C.
13. IBM System/370 Model 155—functional characteristics, pp. 40–44.

Computer Systems

G. Bell, D. Siewiorek, and S.H. Fuller, Editors

Dynamic Memory Repacking

E. Balkovich, W. Chiu, L. Presser, and R. Wood
University of California, Santa Barbara

A probabilistic model of a multiprogramming system is exercised in order to determine the conditions under which the dynamic repacking of main memory is beneficial. An expression is derived for the maximum interference that a repacking process may introduce before the original performance of the system is degraded. Alternative approaches to repacking are discussed, and the operating conditions that lead to improved system throughput through repacking are delineated.

Key Words and Phrases: dynamic memory repacking, memory compaction, storage fragmentation, multiprogramming system model, probabilistic model, central processor productivity, resource utilization, system throughput

CR Categories: 4.32, 5.5, 6.20

1. Introduction

Main memory is one of the most expensive resources in present computer systems. Consequently, much effort has been dedicated to its efficient management. It is well known that main memory fragmentation may prevent the fulfillment of a storage request even though the total available storage exceeds the amount requested. This in turn impacts on the effective utilization of the computer system. A number of solutions have been implemented in an attempt to ameliorate the situation. The principal approaches to memory management have been: repacking, overlaying, paging, segmenting,

This work was supported in part by the National Science Foundation, Grant GJ-31949 and by Delco Electronics. Authors' address: Department of Electrical Engineering and Computer Science, University of California, Santa Barbara, CA 93106.