The use of a synthetic jobstream in performance evaluation

G. Haring,* R. Posch,* C. Leonhardt† and G. Gell†

A new computer system is required to replace the existing UNIVAC 494 system for the universities at Graz. A synthetic jobstream matching the real batch load of the existing system was constructed using Buchholz's job (which was modified and transferred to FORTRAN). This synthetic jobstream was used for comparison of different computer systems as well as for the selection evaluation process itself. The construction of this synthetic jobstream is described in detail in this paper. The problems of characterising the real load are discussed explicitly. Special attention was given to the question of which parameters should be used, together with the decisions on the form of the scale and the subdivision of the range of the parameter values. Furthermore a procedure is given for an optimum of determination of the parameter values of the synthetic jobstream.

(Received August 1977)

Die Universitäten von Graz standen vor der Aufgabe, das existierende EDV-System UNIVAC 494 durch ein neues zu ersetzen. Es wurde ein synthetischer Jobstrom konstruiert, der auf dem synthetischen Job von Buchholz basiert, welcher modifiziert und in FORTRAN übertragen wurde. Dieser synthetische Jobstrom wurde sowohl für den Vergleich verschiedener Computersysteme als auch für den Prozeß der Auswahlbewertung selbst verwendet. Die Konstruktion dieses synthetischen Jobstroms wird in diesem Artikel ausführlich beschrieben. Explizit werden die Probleme im Zusammenhang mit der Charakterisierung der realen Arbeitslast diskutiert. Dabei wird besonders Gewicht auf die Frage der Auswahl der zu verwendenden Parameter sowie die Entscheidung über den Maßstab und die Unterteilung der Wertebereiche der einzelnen Parameter gelegt. Im weiteren wird eine Prozedur zur optimalen Bestimmung der Parameterwerte des synthetischen Jobstroms eingegeben.

1. Introduction

The performance of a computer system depends on a complicated interaction and co-operation of different components—the computer hardware, software and user programs—including some stochastic elements. The results of this co-operation are functions which are requested from the system by the user's application program. The performance of the computer system is a result of the effectiveness with which these functions are processed. That means that a mixture of physically and functionally different parts of the system has to be evaluated. In this context it is important to underline that the discussion and evaluation concerning the performance of a computer system can only be based on the special applications which are to run on the system.

The methods for measuring or describing the performance are different and depend on the special goal. Basically there are three purposes of performance evaluation: performance projection, performance monitoring and selection evaluation. The last one, i.e. performance as a criterion for selecting a particular system, is the most frequent case and also the background of this paper. There are of course several other factors which must be taken into consideration but this will not be discussed further because they are described in another paper reporting on the decision process as a whole (Gell *et al.*, 1979).

As mentioned before, the performance of a computer system depends on its special application. It is therefore important to have a critical knowledge of the workload of the future system. We shall consider the case when a system already in use is to be upgraded or replaced by a new one. That means that a real job profile is known and can therefore be analysed and extrapolated. Proceeding on the assumption that there will be no unexpected or rapid change in the users' resource requirements such an extrapolated future workload can become the basis for evaluating the performance of the new system. This assumption is valid even if the main deficiency in the running system is of qualitative nature, e.g. there is no time sharing facility, because it can be assumed that the processing profile will be independent of the kind of processing. The situation however will be quite different, if the shortcoming of the running system is a quantitative one, e.g. not enough computing power to meet all of the users' needs. The special situation—whether the deficiency is a qualitative or a quantitative one or a combination of both—must be known, e.g. from a questionnaire. Depending on the extent of the quantitative lack, it is necessary to reevaluate the future workload by means of an accurate analysis of the users' additional needs. This can be done in different ways: either by extrapolating the requirements specified by the users or by assuming jobs which have not yet been executed on the system. The requirements of these jobs can be estimated on the basis of defined and specified problems, their solutions and the known users' behaviour. Normally a combination of both methods will provide acceptable results as far as future workload is concerned.

The users of the universities at Graz run their jobs on a UNIVAC 494 with practically no TS-processing and only limited software facilities. Now this system is to be replaced by a new one with qualitative improvements in TS-processing and software supply. An enquiry has shown that there is no demand for extremely positive changes in the quantity of the supplied computing power for the moment. As a consequence of the assumption made above we were able to say that today's resource requirements could be the base of estimation of the future workload. We described this workload, which is a pure batch workload, by different parameters. These parameters formed the basis for constructing a synthetic jobstream

*Technical University of Graz, Steyrergasse 17, A 8010, Graz, Austria
†University of Graz, Universitätsplatz 3, A 8010, Graz, Austria

representative of the real load which served as a means for evaluating the new system. Difficulties arose from the fact that we also intended to test the time sharing (TS) ability of the new system, but we had no information of the future TS workload. A description of the whole test and evaluation process together with a practicable solution for this situation will be published in Posch *et al.* (1979).

2. Evaluation techniques

There is a series of well known techniques for evaluating the performance of a computer system which are listed and defined below:

1. *Timings*
The cycle time and add times are compared to obtain a rating of the evaluated systems.

2. *Instruction mixes*
With this technique the frequency of execution of certain instruction types is specified and timings are weighted accordingly to produce a figure of merit.

3. *Kernel programs*
A kernel program is a typical program to solve a given problem. Then this program has to be timed. The timing can be found either by the execution of the kernel program or usually by an estimation based on the execution times stated by the manufacturer for the instructions that compose the kernel program for a given system. As a rule there are no I/O operations included in a kernel program because the asynchronous nature of the I/O paths make reliable timing difficult.

4. *Analytic models*
An analytic model is a mathematical representation of a computing system, very often based on queueing theory (Kleinrock, 1976).

5. *Benchmarks*
A benchmark is an existing user program, which—together with other selected benchmarks—is executed and timed on the system being tested.

6. *Synthetic jobstream*
A synthetic jobstream consists of synthetic programs, in which the resource requirements of each job are the result of values of special parameters. Synthetic programs include I/O operations and are in fact executed on the system to be evaluated.

7. *Simulation*
In this case the operation of the system is simulated by a special driver which is either based on an event oriented simulation model or on empirically derived data.

It is known and well discussed (Lucas, 1971; Osswald, 1973), that in connection with the selection process only three of these methods are full enough to give satisfactory results. These are the techniques 5, 6, and 7, mentioned above. Perhaps the greatest drawback of simulation is the relatively high costs, for the preparation as well as adaptation and execution. Apart from this fact, simulation creates some other problems not always easy to solve—for instance the decision of how many items should be included in the simulation, the validation of the system or the question of whether all effects of software, such as multiprogramming, time sharing, etc. are adequately considered.

Both of the remaining techniques: benchmarks and synthetic jobstreams, share some common features:

1. The jobs must really be executed by the entire system; therefore all its relevant aspects can be considered and complex situations can be tested. The results can not be refused or denied.
2. The effective processing time or the throughput rather than the actual processor time are measured.

The main problem with both methods is that the resulting workload for the test run must be representative of the job mix of the installation in all the essential parameters of the real workload, which are also to be used in formal description. With the benchmark method this problem is reduced to the proper selection of jobs of the real workload which can either be done randomly or according to a given strategy. The arising difficulties are manifold, mainly because it is necessary to look over a rather long period of time in order to smooth seasonal fluctuations in the workload. If the benchmarks are selected by applying a random number generator based selection process to the sequence of jobs in the real workload during a long period of time, it is not certain that the selected programs are still available—especially in a university environment. There are other problems too, like the translation of a large number of programs into the language used by the system to be tested and questions concerning security and privacy of programs and data, etc. On the other hand, if you take programs based on defined problems instead of extracting jobs from the real workload you will run the risk that the benchmark may not be typical of the existing job profile.

Because of these difficulties we used the synthetic job method for constructing our test run. Compared with the benchmark method this method provides full flexibility since it is possible to design jobs which can include almost any desired parameters for measurement. The whole test run can be built as a sequence of formally identical programs in which the resource requirements of each single job are determined by special job parameters (see Section 3). In this case the main problem is how to represent and describe the real job profile, that means to decide upon the variables used for describing the resource requirements of each job in the real workload and the scale and subdivision for these variables, etc. The second point in the process of describing a real workload by means of a synthetic jobstream concerns the connection between this real workload and the synthetic jobstream by determining the special characteristics of each synthetic job. A description of these problems as well as a practical solution to them will be found in this paper.

3. Connection between real and synthetic workload

This chapter describes the method used to obtain a representative synthetic workload from a real job profile. The transition from the real to the synthetic workload can be done in two steps:

1. Find out the characteristics of the real workload.
2. Fit the synthetic jobstream to these characteristics.

The method for this adaptation follows a paper of Sreenivasan and Kleinman (1974) and is based on the joint probability density of the real workload which is matched by that of the synthetic stream. Each real job of the workload makes demands on the various system resources, i.e. each of the real jobs can be described by n variables $X_1, X_2, \ldots, X_n$, which represent characterising parameters of the jobs which are loaded. The values of these variables which are known from the system logfile for each job provide a measure of the usage of the different system resources by this job. Thus each job can be seen as a point in an n-dimensional space. The whole workload is represented by a set of points in this space. But an amorphous mass of points makes no sense. One possibility for grouping this set consists in using cluster methods for finding groups of jobs with special requirements of the system resources (Hunt *et al.*, 1971). Another method is to calculate the

joint probability density of jobs in discrete cells of the space. Being aware of the fact that both methods are adequate for constructing a synthetic workload, we decided to use the second because it consumes less processing time. By dividing each coordinate axis i ($i = 1, 2, \ldots, n$) into l_i discrete but not necessarily equal intervals the whole space was subdivided into $\prod_{i=1}^{n} l_i$ discrete cells. With N_{tot} representing the total number of jobs in the real workload and

$$N_{i_1 i_2 \ldots i_n} (1 \leqslant i_r \leqslant l_r, r = 1, 2, \ldots, n)$$

the number of jobs in the cell ($i_1\ i_2 \ldots i_n$), the probability of finding a job in this cell is expressed by

$$P_{i_1 i_2 \ldots i_n} = N_{i_1 i_2 \ldots i_n} / N_{tot} \tag{1}$$

Using these values of $P_{i_1 i_2 \ldots i_n}$, the values of $N'_{i_1 i_2 \ldots i_n}$ (primed quantities refer to the synthetic jobstream) for the synthetic jobstream are obtained from

$$N'_{i_1 i_2 \ldots i_n} = P_{i_1 i_2 \ldots i_n} \cdot N'_{tot} \tag{2}$$

where N'_{tot} is the assumed total number of jobs in the synthetic workload.

The real total number of jobs in the synthetic jobstream can be obtained by summing $N'_{i_1 i_2 \ldots i_n}$ over all cells taking into account that the right side of equation (2) must represent an integer value. By means of this equation the joint probability density of the synthetic stream can be matched with that of the real workload.

Within the synthetic jobstream it is necessary to characterise the $N'_{i_1 i_2 \ldots i_n}$ jobs of the cell ($i_1\ i_2 \ldots i_n$) by values for these n variables. In contrast to Sreenivasan and Kleinman (1974), who used the central point of the interval as a representative value for each variable, we assumed that we could obtain a better representation of the jobs in each cell by using the mean value $\bar{x}_j$ of each variable X_j ($j = 1, 2, \ldots, n$) taken from the $N_{i_1 i_2 \ldots i_n}$ jobs of the real workload:

$$\bar{x}_j(i_1\ i_2 \ldots i_n) = \frac{1}{N_{i_1 i_2 \ldots i_n}} \cdot \sum_{v=1}^{N_{i_1 i_2 \ldots i_n}} x_j^{(v)}(i_1\ i_2 \ldots i_n); \quad j = 1, 2, \ldots, n \tag{3}$$

where $x_j^{(v)}(i_1\ i_2 \ldots i_n)$ is the value of the variable X_j for the v-th job belonging to the cell ($i_1\ i_2 \ldots i_n$). Therefore the $N'_{i_1 \ldots i_n}$ jobs of the synthetic jobstream representing the $N_{i_1 i_2 \ldots i_n}$ jobs of the real workload belonging to the cell ($i_1\ i_2 \ldots i_n$) have the identical requirements of system resources which can be determined from the characterising values $\bar{x}_j(i_1\ i_2 \ldots i_n)$ with $j = 1, 2, \ldots, n$.

For the variable X_k the total requirement consumed by the whole synthetic stream is given by:

$$\mathrm{tot}(x_k) = \sum_{i_1=1}^{l_1} \sum_{i_2=1}^{l_2} \ldots \sum_{i_n=1}^{l_n} \bar{x}_k(i_1\ i_2 \ldots i_n) \cdot N'_{i_1 i_2 \ldots i_n}; \quad k = 1, 2, \ldots, n \tag{4}$$

This equation can be used to determine N'_{tot}, a method which is described later in this paper.

The method of matching the joint probability distribution favours the jobs that occur most frequently. Often there are situations where infrequent jobs place heavy demands on the system resources. In this case the considerations made above are altered by using a weighted joint probability density (Sreenivasan and Kleinman, 1974). This modification was not used in our situation, as will be explained later in this paper.

4. Characteristics of the real workload

To obtain the characteristics of the jobs running on the UNIVAC 494 we evaluated the system logfiles during the period from 1 January to 30 June. A total of 51716 jobs was executed, 30352 of them with at least one FORTRAN task and 44837 with at least one execution task. The major part of those programs having no FORTRAN task but at least one GO task were relocatable elements resulting from a FORTRAN compilation with a following load task. Therefore it seemed obvious to reconstruct the real batch workload from synthetic FORTRAN jobs. The form of these synthetic FORTRAN programs will be described in Section 6.

To characterise the jobs of the real workload we selected three variables:

X_1 Total number of words transferred between central memory and second storage (discs and drums) and vice versa.

X_2 Total CPU time used by the execution phases of the job.

X_3 Maximum amount of core used by the execution phases of the job.

An important decision was to choose the special scale and intervals of the three describing variables resulting in a subdivision of the space into cells. In this special case we considered linear and logarithmic scales. For variables X_1 and X_2 we decided on a logarithmic scale, with parameter X_3 we used a linear scale. The logarithmic scale should guarantee that those jobs which occur infrequently but use resources of the system intensively are adequately considered. Otherwise these jobs would either not be represented properly in the synthetic jobstream or small differences in the number of jobs in these weakly populated cells would result in assignments of synthetic jobs to single cells not representing the behaviour of the job profile of this cell. Together with the decision on the form of the scale we decided on the upper limits of the three variables. 32K was a natural limit for the core requirement because on the UNIVAC 494 FORTRAN generated programs can only be addressed within this limit. As an upper limit for X_1 we took 100 megawords and for X_2 3020 secs resulting in the fact that only one of about 1000 jobs did not fall in a cell of our three-dimensional space.

In order to decide on the number of intervals l_i for the values of each variable a model of the synthetic jobstream was generated by means of a program which calculated the mean values of the variables and their ratios using different values of N'_{tot} and different subdivisions of the parameter axis. This decision is strongly connected with the stability behaviour of the selected subdivision. The definition of the stability of a given subdivision is based on the stability of a range of N'_{tot}. The stability of an interval in which N'_{tot} ranges is defined by the length of this interval divided by the sum of the differences between the smallest and highest values of each of the [0, 1]-normalised parameters in this interval. The stability of a subdivision is defined as the maximum stability of an interval with a predefined length.

The term 'execution phase of the job' comprehends all 'GO' tasks of a job but no compilations or other phases.

In addition to these variables the jobs were characterised by the following parameters, which were however not considered as separate variables in the sense of Section 3:

1. Ratio of the number of words transferred to the disc subsystem to the number of words transferred to the drum subsystem: this ratio was used for tests on the UNIVAC 494 only because our equipment comprised these subsystems. In the synthetic jobstream for evaluating other systems all transfers were made to a disc subsystem.

2. Number of printed lines: originally it was intended to use this parameter as a fourth dimension. Later this idea was rejected because the I/O transfers resulting from the print file of the job were already taken into account in variable X_1 and a fourth dimension would have caused troubles in

dimensioning the program system to analyse the real workload. In each cell (i_1 i_2 i_3) the mean value of the number of lines printed by jobs in this cell was therefore calculated and used as a parameter in the synthetic program.

3. Time of compilation and number of FORTRAN statements: the mean values of these two parameters were calculated for each cell to see whether it was justified to assume that there was a strongly positive correlation between these two parameters and that we could confine ourselves to the number of FORTRAN statements to reconstruct the compilation part of the real workload.
4. Number of cards read: this parameter was neglected because the major part of cards read—the program cards—were already considered in the number of FORTRAN statements and the number of data cards was small in comparison to these. Large data files are usually read from a secondary storage and are therefore considered in parameter X_1.

The following three subdivisions were evaluated:

(*a*)	X_1: 0-100	MCW	10 intervals
	X_2: 0-3020	sec	8 intervals
	X_3: 0-32	KCW	4 intervals
(*b*)	X_1: 0-100	MCW	10 intervals
	X_2: 0-3020	sec	8 intervals
	X_3: 0-32	KCW	8 intervals
(*c*)	X_1: 0-100	MCW	20 intervals
	X_2: 0-1800	sec	15 intervals
	X_3: 0-32	KCW	8 intervals

The subdivision (*a*) had a relatively stable behaviour but as a result of the large core intervals—consequence of the rounding errors in equation (2)—too many memory-intensive jobs had to be considered in the synthetic jobstream which led to a mean value for memory requirements which was substantially higher than the mean value of those in the real workload. As a consequence of the relatively small cells subdivision (*c*) was very instable. No stability could be reached within an acceptable total amount of CPU time for the synthetic jobstream. The

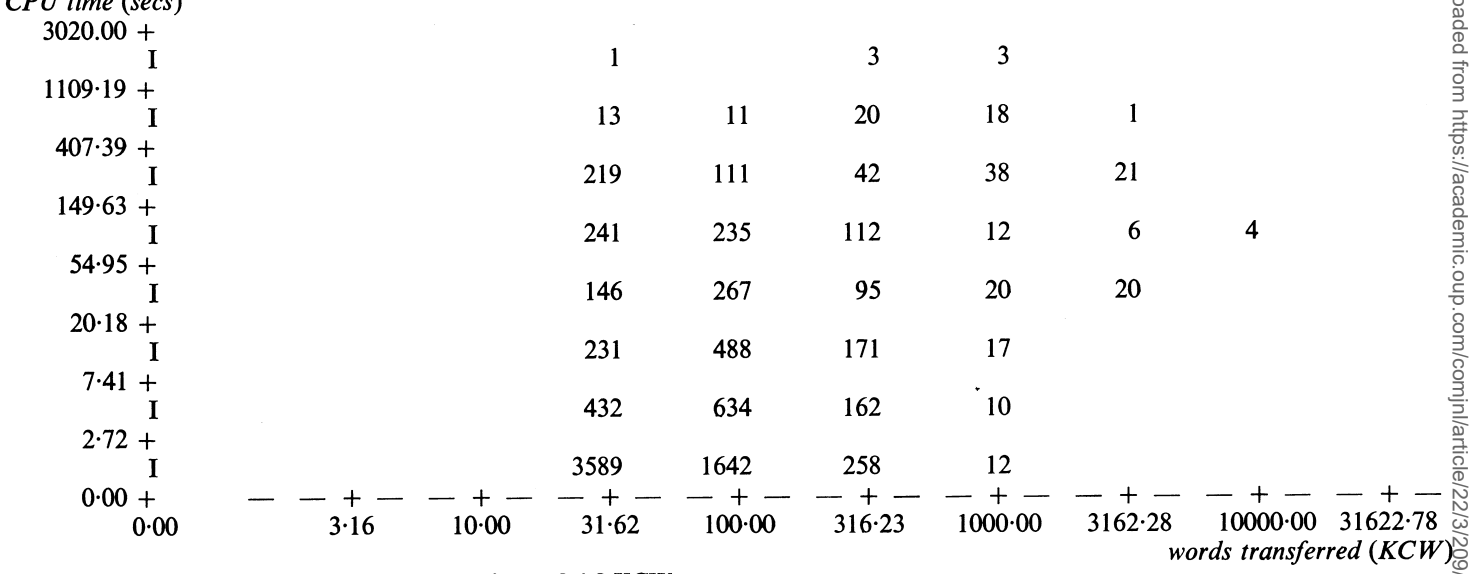

Fig. 1 Distribution of the real jobs in the core interval 4-8 KCW

number of jobs		*CPU time (secs)*		*central memory (KCW)*		*words transferred (KCW)*		*CPU time/ words transf.*
in	*effect*	*total*	*mean*	*total*	*mean*	*total*	*mean*	
250	227	7327·7	32·3	2664·4	11·7	28725·0	126·5	3·92
255	231	7590·6	32·9	2707·0	11·7	29052·0	125·8	3·83
260	242	7979·7	33·0	2832·4	11·7	32068·0	132·5	4·02
265	247	8028·6	32·5	2879·4	11·7	32639·0	132·1	4·07
270	250	8030·7	32·1	2897·6	11·6	32773·0	131·1	4·08
275	256	8348·3	32·6	2993·6	11·7	38859·0	151·8	4·65
280	262	9120·9	34·8	3105·4	11·9	41433·0	158·1	4·54
285	268	9454·2	35·3	3184·7	11·9	59938·0	223·6	6·34
290	272	9548·6	35·1	3201·7	11·8	60127·0	221·1	6·30
295	278	9652·2	34·7	3258·7	11·7	60814·0	218·8	6·30
300	281	9654·6	34·4	3306·0	11·8	60923·0	216·8	6·31
305	287	10314·8	35·9	3405·3	11·9	79624·0	277·4	7·72
310	293	10645·8	36·3	3446·9	11·8	80063·0	273·3	7·52
315	298	10893·9	35·9	3522·4	11·8	80835·0	271·3	7·56
320	304	11666·1	38·4	3596·1	11·8	83725·0	275·4	7·18
325	310	11727·0	37·8	3681·6	11·9	84064·0	271·2	7·17
330	315	11772·5	37·4	3757·0	11·9	84363·0	267·8	7·17
335	318	11941·7	37·6	3796·2	11·9	84590·0	266·0	7·08
340	323	11952·3	37·0	3847·7	11·9	84777·0	262·5	7·09
345	329	12310·2	37·4	3944·8	12·0	85076·0	258·6	6·91
350	331	12310·8	37·2	3957·6	12·0	85113·0	257·1	6·91

Fig. 2 Short cut of the output to determine N'_{tot}

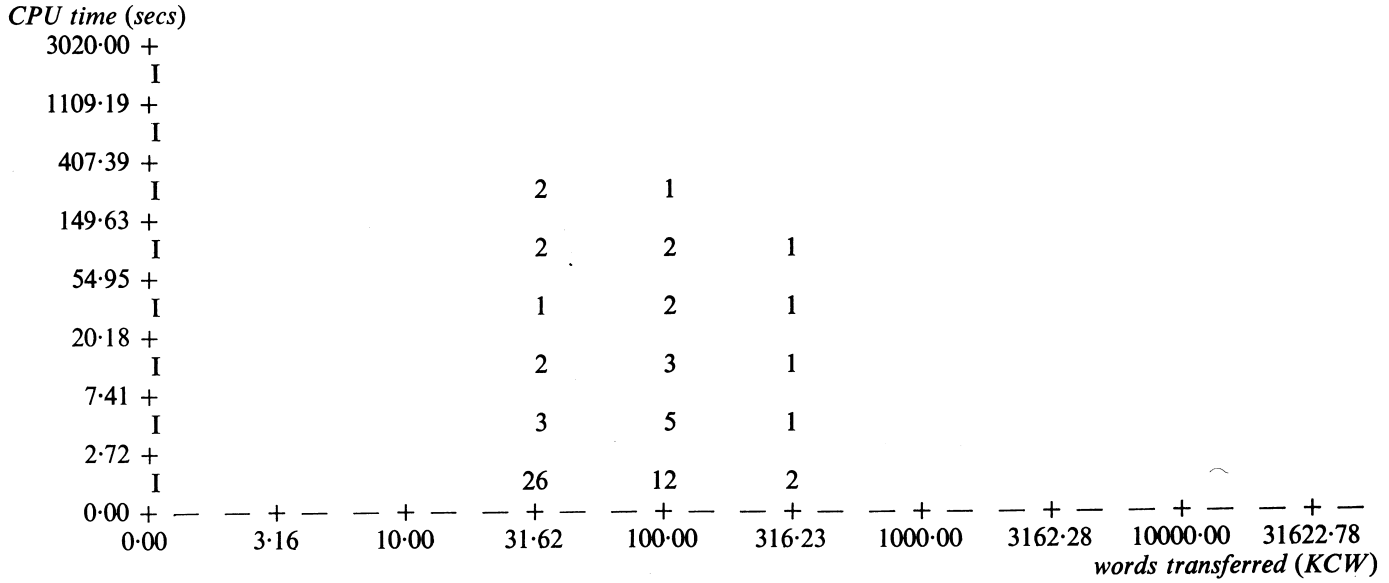

Fig. 3 Distribution of the synthetic jobs in the core interval 4-8 KCW

most acceptable subdivision was (*b*), resulting in a joint probability density which could be used for constructing the synthetic jobstream. Its distribution in the core interval 4-8 KCW is shown as an example in **Fig. 1.**

5. Number and distribution of jobs in the synthetic stream

With subdivision (*b*) described in Section 4 we had to determine N'_{tot} the number of jobs in the synthetic stream. For this purpose we determined $N'_{i_1 i_2 \ldots i_n}$ from equation (2) by means of the joint probability density calculated from equation (1) and the total requirements of the whole synthetic jobstream according to equation (4). This procedure was done for N'_{tot} between 50 and 800 incremented by 5. In addition to the total requirements the mean values of the three variables X_1, X_2, X_3 and the ratio X_2/X_1 were also calculated. **Fig. 2** gives a short cut of the resulting output. The final choice of N'_{tot} was guided by the following considerations:

1. The mean values of the variables for the synthetic load should be within a reasonable neighbourhood of those of the real workload. The expected difference between these values and those of the real load was of 25% approximately for X_1 and X_2 because infrequent jobs with heavy demands on the resources of the system still occurred but were not represented in the synthetic stream. We therefore tried to find a good coincidence for the ratio X_2/X_1 of the real and synthetic loads, thus ensuring that the synthetic jobstream has the same structure of CPU/I/O usage as the real load.
2. The total CPU time used by the synthetic stream should not be higher than a given value (4 hours on the UNIVAC 494).
3. N'_{tot} should be determined in such a way that there were only stable intervals beyond this value.

Considering these three points we selected $N'_{tot} = 320$ as the total number of jobs which corresponds to an effective stream of 304 jobs. The difference of 16 jobs results from rounding in equation (2) because it is impossible to assign, for instance, 2·4 jobs to a cell.

	real load	*synthetic load*
number of jobs	44,837	304
central memory (mean)	12·4 KCW	11·8 KCW
CPU-time (mean)	48·4 sec	38·4 sec
words tranferred (mean)	370·4 KCW	275·4 KCW
CPU-time/words transf.	7·66	7·18

Fig. 4 Comparison between real and synthetic load ($N'_{tot} = 320$)

Fig. 3 shows the distribution of the synthetic jobs in the core interval 4-8 KCW for comparison with Fig. 1. The requirements to be matched from these jobs in each cell are known from the real workload by equation (3). **Fig. 4** shows a comparison between the different values of the real and synthetic workload. By equation (4) the total of the expected CPU time of the whole synthetic jobstream was 11666 secs.

6. The synthetic job

As already mentioned in Section 4 we used a synthetic FORTRAN program to build a jobstream, because FORTRAN is the programming language primarily used on our system. We restricted ourselves to one program, which means that the total synthetic jobstream is built on the basis of FORTRAN programs in which each program consists of a sequence of identical statements. This restriction to one program was justified by the fact that the parameters used in this synthetic program were sufficient to describe the real workload.

In Section 4 we pointed out that the number of FORTRAN statements showed a strongly positive correlation to the compilation time (correlation coefficient = 0·92). Therefore we restricted our synthetic program to 288 program cards—the mean value of program cards per job in the real workload—to reconstruct the compilation part of the real load.

The basis for the synthetic program was Buchholz's synthetic job (1969) which was transformed to FORTRAN. Doing this transformation the following points had to be considered to obtain a version acceptable for the predefined goals:

1. The program has to be written in ANSI standard FORTRAN to prevent problems when using this program on different systems.
2. The FORTRAN version should consist of 288 statements exactly.
3. The structure of the original version of Buchholz's job should be preserved.
4. In addition to the original version it should be possible to simulate the print parts of programs.
5. The CPU time shall be consumed primarily by statements, the distribution of which corresponds to a workload, resulting from scientific applications.

The synthetic program consists of the following parts:

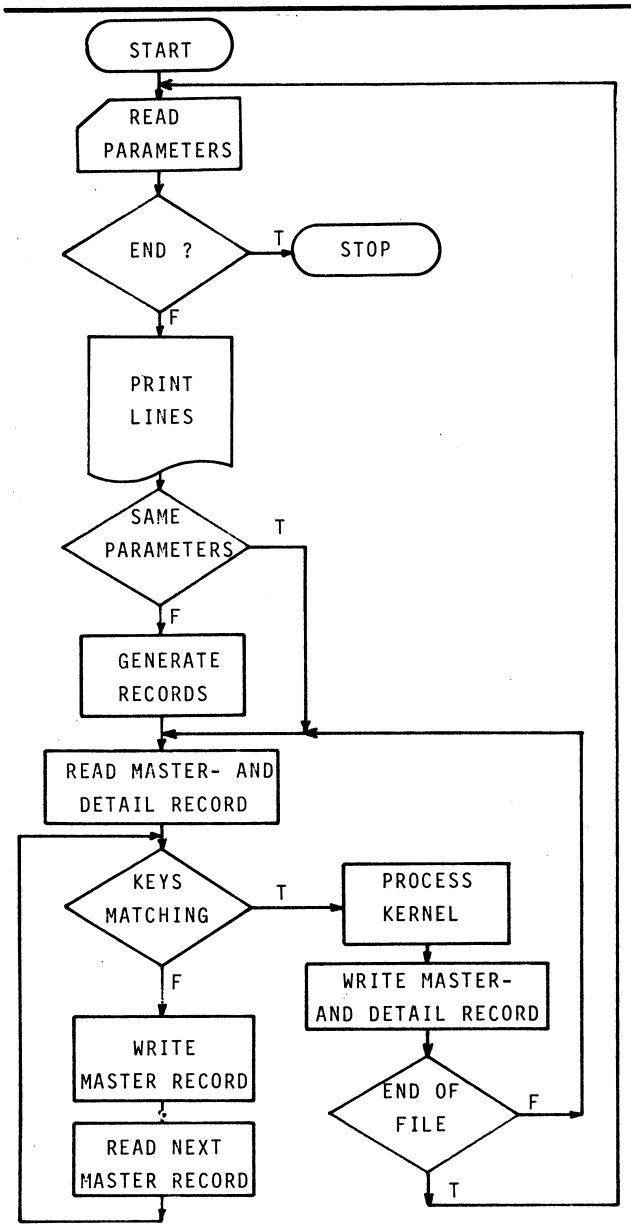

Fig. 5 Overall flowchart of the synthetic program

declaration, print, file generation, file and kernel processing. **Fig. 5** shows an overall flowchart of the program. Usually the program consists of generation of a file of master records and one of detail records on a secondary storage according to the parameters read. The records of both files are then read and compared. If they match, the kernel of the program will be processed. If not the next master record will be read. This part includes the generation of a new master and a new detail file. **Fig. 6** gives a listing of our FORTRAN version of the program.

Within the declaration part the only system-dependent assignments reside in the DATA statements specifying the logic unit number for the reader (ICIN = 1), the printer (IP = 2) and the secondary storage units (MASGEN = 5, DETGEN = 6, NEWMAS = 7, DETOUT = 8).

As mentioned before the requirements of a single synthetic job were determined by the parameters describing the program. The memory requirement of the job could be fitted by choosing the dimension of the array TABLE accordingly (see Section 7). The selected value of the dimension has also to be assigned to the variable IDIM and to be used in the dimension of TABLE 1 which occupies the same memory locations as TABLE. For each synthetic job there are four integer parameters, the values of which have to be read from data cards:

NMAS number of master records to be generated

```
*JOB F TUG/RACHHOLZ J,F8050RJE01,,15D,100D/0
$
*FOR NY SYN
$      INTEGER TABLE1(_________,1),TABLE(_____)
       INTEGER U,CHECK,COUNT,CARD(20),TEMPC,SUM,
      *          KRET,DREC(51),DKEY(3),DSUM,DCHECK,DDATA(3,15),
      *MASGEN,MASTER,DETOUT,DETGEN,DETIN
       LOGICAL ILOG,LOGEND
       DIMENSION MREC(51),MKEY(3),MDATA(3,15), ZBUF(6),INTEG(35)
       DIMENSION ITEXT(15)
       EQUIVALENCE(MREC(1),MKEY(1)),
      *           (MREC(4),MSUM),
      *           (MREC(5),MCHECK),
      *           (MREC(6),MDATA(1,1))
       EQUIVALENCE(DREC(1),DKEY(1)),
      *           (DREC(4),DSUM),
      *           (DREC(5),DCHECK),
      *           (DREC(6),DDATA(1,1))
       EQUIVALENCE(MASGEN,MASTER),
      *           (DETGEN,DETIN)
       EQUIVALENCE(CARD(3),NMAS1),
      *           (CARD(5),NDET1),
      *           (CARD(7),NREP),
      *           (CARD(9),NPRINT)
       EQUIVALENCE(TABLE(1),TABLE1(1,1))
       DATA DET1,DET2/4HDETA,4HIL  /
       DATA MAST1,MAST2/4HMAST,4HER  /
       DATA INULL,ISL,IZZ/4H0000,4H////,4HZZZZ/
       DATA IPAS/4H PAS/
       DATA ICIN,IA,IB,SUM/1,253,17,25365/
       DATA IP/2/
       DATA MASGEN,DETGEN,NEWMAS,DETOUT/5,6,7,8/
       TIME(I)=I
       ILOG=.TRUE.
       IZ=1
       ELTIME=0.
       NMAS=-1
       NDET=-1
$      IDIM=_____
C
       DO 10 J=1,IDIM
   10  TABLE(J)=J-1
       DO 1000 I=1,15
 1000  ITEXT(I)=INULL
       DO 1001 I=1,6
 1001  ZBUF(I)=1725.13
       DO 1002 I=1,35
 1002  INTEG(I)=I-1
    1  READ(ICIN,100) (CARD(I),I=1,20)
       LOGEND=.FALSE.
       IF(CARD(1).EQ.ISL) GOTO 16
       WRITE(IP,207) NMAS1,NDET1,NREP,NPRINT
       IF(CARD(1).NE.IPAS) GOTO 1
       IF(NPRINT.EQ.0) GOTO 17
       DO 151 I=1,NPRINT,3
       WRITE(IP,208) (INTEG(J),J=1,35)
       WRITE(IP,209) (ITEXT(J),J=1,15)
  151  WRITE(IP,206) (ZBUF(J),J=1,6)
   17  CONTINUE
       COUNT=0
       CHECK=0
C
C      MASTER GENERATION
C
    2  IF(NMAS1.EQ.NMAS) GOTO 3
       NMAS=NMAS1
       REWIND MASGEN
       IF(NMAS.EQ.0) GOTO 211
       DO 20 J=1,NMAS
       MSUM=0
       J1=J/100000
       J2=J/10000-10*J1
       J3=J/1000-10*J2-100*J1
       J4=J/100-10*J3-100*J2-1000*J1
       J5=J/10-10*J4-100*J3-1000*J2-10000*J1
       J6=J-J5*10-J4*100-J3*1000-J2*10000-J1*100000
       INTKEY=J6+J5*16+J4*256+J3*4096+J2*65536+J1*1048576
       MKEY(1)=INULL
       MKEY(2)=INULL
       MKEY(3)=INTKEY
       CHECK=CHECK+J
       MCHECK=CHECK
       TEMPC=INTKEY
       DO 21 K=1,15
       MDATA(1,K)=MAST1
       MDATA(2,K)=MAST2
   21  MDATA(3,K)=TEMPC
   20  WRITE(MASGEN) MREC
  211  CHECK=0
       MREC(1)=ISL
       WRITE(MASGEN) MREC
C
C      DETAIL GENERATION
C
    3  IF(NDET1.EQ.NDET) GOTO 4
       NDET=NDET1
       IRATIO=NMAS/NDET
       REWIND DETGEN
       IDANZ=1
       IF(NDET.EQ.0) GOTO 301
       DO 30 J=IRATIO,NMAS,IRATIO
       DSUM=0
       J1=J/100000
       J2=J/10000-10*J1
```

Fig. 6 Listing of the synthetic program (continued overleaf)

```
      J3=J/1000-10*J2-100*J1
      J4=J/100-10*J3-100*J2-1000*J1
      J5=J/10-10*J4-100*J3-1000*J2-10000*J1
      J6=J-J5*10-J4*100-J3*1000-J2*10000-J1*100000
      INTKEY=J6+J5*16+J4*256+J3*4096+J2*65536+J1*1048576
      DKEY(1)=INULL
      DKEY(2)=INULL
      DKEY(3)=INTKEY
      CHECK=CHECK+J
      DCHECK=CHECK
      TEMPC=INTKEY
      DO 31 K=1,15
      DDATA(1,K)=DET1
      DDATA(2,K)=DET2
   31 DDATA(3,K)=TEMPC
      WRITE(DETGEN) DREC
      IF(IDANZ.EQ.NDET) GOTO 301
   30 IDANZ=IDANZ+1
  301 DREC(1)= ISL
      WRITE(DETGEN) DREC
      CHECK=0
C
C     TAPE PASS
C
    4 IF(NMAS1.LE.0) GOTO 1
      ASSIGN 7 TO KRET
      REWIND MASTER
      REWIND NEWMAS
      REWIND DETIN
      REWIND DETOUT
      READ(MASTER) MREC
      IF(MREC(1).EQ.ISL) GOTO 11
      IF(LOGEND) GOTO 8
      READ(DETIN) DREC
      IF(DREC(1).EQ.ISL) GOTO 9
      T=TIME(0)
C
C     KEYTEST
C
    5 CONTINUE
      IF(MKEY(3).LT.DKEY(3).OR.LOGEND) GOTO 8
      IF(MKEY(3).EQ.DKEY(3)) GOTO 53
      WRITE(IP,200)
      GOTO 12
   53 CONTINUE
C
C     KERNEL
C
6     B=100.
      C=7.
      I=0
   61 I=I+1
      IF(I.GT.NREP) GOTO 63
      IF(IZ.GE.IDIM) IZ=1
      ISW=IDIM/NREP
      IF(ISW.EQ.0) ISW=1
      IC=I
      A=20.
      JJ=1
      GOTO 6101
 6102 IF((IA.GT.IB).OR.(I.GT.0).OR.ILOG) GOTO 6103
 6104 IF((IA.LE.IB).OR.(IZ.NE.0)) GOTO 6105
 6106 IF((IA.GT.IB).OR.(I.GT.0)) GOTO 6107
 6108 IF(IC) 6105,6106,6109
 6109 IF(11.*A*C+A+B+1.) 6105,6106,6110
 6107 IF(11./33.+A/22.+13/25.) 6105,6106,6108
 6105 IF(-11.*A*C+A+B+1.) 6106,6107,6108
 6103 IF(-257.*A*C-A-B-1.) 6104,6105,6106
 6101 IF(-1.*A*B-A-B-13.) 6102,6103,6104
 6110 IH=IZ+JJ
      TABLE1(IZ,JJ)=TABLE(IZ)+TABLE(IH)
      IA=IC+IA+TABLE(IZ)+IA
      IC=10
      GOTO 611
  612 IC=13
      A=257.
      GOTO 613
  614 IC=25
      A=7.
      ILOG=.FALSE.
      GOTO 615
  616 A=3925.
      B=23.
      GOTO 617
  618 A=9.
      GOTO 619
  620 A=298.
      GOTO 621
  611 A=20.
      B=23.
      GOTO 612
  613 A=2.
      B=7.
      GOTO 614
  615 A=2932.
      B=0.
      GOTO 616
  617 A=15.
      GOTO 618
  619 B=25725.
      GOTO 620
  621 A=71.
      ILOG=.TRUE.
      GOTO 622
  623 B=5.
      GOTO 624
  625 A=3.
      GOTO 626
  627 B=3.
      GOTO 628
  629 A=237.
      B=1.
      IC=5
      GOTO 630
  622 A=9.
      GOTO 623
  624 IC=257
      GOTO 625
  626 B=723.
      GOTO 627
  628 B=13.
      GOTO 629
  630 A=7.
      IZ=IZ+ISW
      GOTO 61
   63 GOTO KRET, (7,15)
C
C     WRITE DETAIL
C
    7 NSUM=SUM
      DSUM=SUM
      CHECK=DCHECK
      COUNT=COUNT+1
      WRITE(DETOUT) DREC
      READ(DETIN) DREC
      IF(DREC(1).EQ.ISL) GOTO 9
C
C     WRITE MASTER
C
    8 WRITE(NEWMAS) MREC
      READ(MASTER) MREC
      IF(MREC(1).EQ.ISL) GOTO 11
      GOTO 5
C
C     RUNOUT
C
    9 LOGEND=.TRUE.
      DKEY(3)=IZZ
      WRITE(DETOUT) DREC
      GOTO 8
C
C     END TPASS
C
   11 ELTIME = TIME(1)
      WRITE(NEWMAS) MREC
      IF(CHECK.EQ.(COUNT*(COUNT+1)*IRATIO)/2)GOTO 12
      WRITE(IP,203)
   12 CONTINUE
C
C     PRT END
C
   13 WRITE(IP,204) ELTIME
      WRITE(IP,202) COUNT
      GOTO 1
C
C     COMPUTE PASS
C
   14 ASSIGN 15 TO KRET
      T=TIME(0)
      GOTO 6
C
C     CRETURN
C
   15 ELTIME=TIME(1)
      GOTO 13
   16 STOP
  100 FORMAT(2A4,4(I6,A2),10A4)
  200 FORMAT(//26H SEQUENCE ERROR HALTED RUN  )
  202 FORMAT(5X,15HAKTIVE RECORDS: ,I6)
  203 FORMAT(//27H CHECKSUM ERROR HALTED PASS )
  204 FORMAT(/15H ELAPSED TIME=  ,F7.1)
  206 FORMAT(1X,6F10.2)
  207 FORMAT(1X,6HNMAS1=,I6,1H ,6HNDET1=,I6,1H ,5HNREP=,I6,2H
     *7HNPRINT=,I6)
  208 FORMAT(1X,10I1,25I2)
  209 FORMAT(1X,15A4)
      END
*END
```

Fig. 6 (continued)

NDET number of detail records
NREP number of repetitions on processing the kernel
NPRINT number of lines to be printed.

It was also necessary to provide the possibility of reading not only one data card but NPAS data cards because with big jobs the values of NMAS, NDET, etc. could be too large, leading to an integer overflow or the reservation of too large areas on the secondary storage. The total requirements caused by the parameter values of the NPAS data cards should correspond to the prescribed resource requirements of each job. The procedure for determining NPAS and the parameter values is described in Section 8.

Within the file generation part of the program NMAS master records and NDET detail records were generated and written on a secondary storage file together with a trailer record for each file. Each output statement transfers an unformatted FORTRAN block (265 CW on a UNIVAC 494) from the central memory to the secondary storage. Each of these blocks represents a record consisting of an integer array of 51 elements. The numbering of the records was done by simulating the PL/I integer to text conversion based on the conversion of the decimal value into a new value with a hexadecimal basis. In this way the master records were numbered from 1 to NMAS in the decimal system. The detail records were given keys between 1 and NMAS, in which a step of ⌊NMAS/NDET⌋ was used. This system made it necessary that NMAS $\geqslant$ NDET.

The kernel of the synthetic job, which is executed NREP times if the key of the master record matches with the key of the detail record, consists of instructions the distribution of which follows the Gibson Mix II for scientific applications (Osswald, 1973). The kernel is primarily responsible for the CPU time consumption of the job. The memory references of the kernel are distributed in a cyclic way to prevent a system with paging facilities from swapping parts of the data area for a long time. Arithmetic overflows occurring on special systems should be ignored.

7. The characteristic parameters

The parameter values of each job determine the resource requirements of this job. These are in particular the total area of central memory used, the total CPU time used, the number of words transferred and the number of lines printed by this job. For reconstructing the real workload, it is important to know the relation between the parameter values and the resulting resource requirements.

The amount of central memory used results directly from the values of the dimension for the array TABLE

$$\mathrm{CORE}_{\mathrm{total}} = \mathrm{CORE}_{\mathrm{basic}} + \text{dimension of TABLE} \qquad (5)$$

where $\mathrm{CORE}_{\mathrm{basic}}$ represents the core requirements of the program without the array TABLE together with all linked routines including the runtime system (4030 CW on the UNIVAC 494).

The total number of lines printed results from the sum of the different values for NPRINT on the NPAS data cards:

$$\mathrm{NPRINT}_{\mathrm{total}} = \sum_{i=1}^{n} \mathrm{NPAS}_i \,.\, \mathrm{NPRINT}_i \qquad (6)$$

where we assume that there are n different groups of data cards, each group consisting of NPAS_i identical cards. If there is only one set of identical data cards, i.e. $n = 1$, this relation reduces to

$$\mathrm{NPRINT}_{\mathrm{total}} = \mathrm{NPAS} \,.\, \mathrm{NPRINT} \qquad (6')$$

where NPAS represents the total number of data cards and NPRINT the value for the number of lines to be printed which appears on each of the NPAS data cards.

The total amount of CPU time used results from

$$\mathrm{CPTIME}_{\mathrm{total}} = a_0 + \sum_{i=1}^{n} (a_1 \,.\, \mathrm{NPAS}_i + a_2 \,.\, (\mathrm{NMAS}_i + \mathrm{NDET}_i) + a_3 \,.\, \mathrm{NPAS}_i \,.\, (\mathrm{NMAS}_i + \mathrm{NDET}_i) + a_4 \,.\, \mathrm{NDET}_i) \,.\, \mathrm{NREP}_i \,.\, \mathrm{NPAS}_i + a_5 \,.\, \mathrm{NPAS}_i \,.\, \mathrm{NPRINT}_i) \qquad (7)$$

The terms in this equation correspond to the different steps in the synthetic program:

a_0	the CPU time used if all parameters are equal to zero
$a_1 \,.\, \mathrm{NPAS}_i$	results from the handling of the data cards
$a_2 \,.\, (\mathrm{NMAS}_i + \mathrm{NDET}_i)$	results from the generation of the master and detail files
$a_3 \,.\, \mathrm{NPAS}_i \,.\, (\mathrm{NMAS}_i + \mathrm{NDET}_i)$	CPU time used for processing and updating files
$a_4 \,.\, \mathrm{NDET}_i \,.\, \mathrm{NREP}_i \,.\, \mathrm{NPAS}_i$	CPU time spent in the kernel of the program
$a_5 \,.\, \mathrm{NPAS}_i \,.\, \mathrm{NPRINT}_i$	CPU time used for printing lines.

It was not necessary to consider the trailer blocks explicitly, because in the first case (generation of files) there are exactly two trailer blocks, the influence of which is therefore covered by the term a_0. In the second case (processing of files) there are exactly two trailer blocks for each execution following NPAS_i which are therefore covered by the term $a_1 \,.\, \mathrm{NPAS}_i$.

In our situation ($n = 1$), this equation reduces to

$$\mathrm{CPTIME}_{\mathrm{total}} = a_0 + a_1 \,.\, \mathrm{NPAS} + a_2 \,.\, (\mathrm{NMAS} + \mathrm{NDET}) + a_3 \,.\, \mathrm{NPAS} \,.\, (\mathrm{NMAS} + \mathrm{NDET}) + a_4 \,.\, \mathrm{NDET} \,.\, \mathrm{NREP} \,.\, \mathrm{NPAS} + a_5 \,.\, \mathrm{NPAS} \,.\, \mathrm{NPRINT} \qquad (7')$$

Like the CPU time, the number of transferred words can be derived from

$$\mathrm{TRANS}_{\mathrm{total}} = b_0 + \sum_{i=1}^{n} b_1 \,.\, (2 \,.\, \mathrm{NPAS}_i + 1) \,.\, (\mathrm{NMAS}_i + \mathrm{NDET}_i + 2) \,.\, 265 + b_2 \,.\, \mathrm{NPAS}_i \,.\, \mathrm{NPRINT}_i \qquad (8)$$

where the different terms represent the following:

b_0	the amount of transfers if there are no explicit file or printer operations
$b_1 \,.\, (2 \,.\, \mathrm{NPAS}_i + 1) \,.\, (\mathrm{NMAS}_i + \mathrm{NDET}_i + 2) \,.\, 265$	number of words by file operations where the number of blocks results from the sum of NMAS master records, NDET detail records and two trailer blocks, which were written once when being generated and were then read and written during processing, i.e. during updating the files for each data card (NPAS_i)–giving a factor of $(2 \,.\, \mathrm{NPAS}_i + 1)$. Each block has 265 CW.
$b_2 \,.\, \mathrm{NPAS}_i \,.\, \mathrm{NPRINT}_i$	number of transfers resulting from printing output lines.

For $n = 1$ the equation reduces to

$$\mathrm{TRANS}_{\mathrm{total}} = b_0 + b_1 \,.\, (2 \,.\, \mathrm{NPAS} + 1) \,.\, (\mathrm{NMAS} + \mathrm{NDET} + 2) \,.\, 265 + b_2 \,.\, \mathrm{NPAS} \,.\, \mathrm{NPRINT} \qquad (8')$$

Estimators a_i^*, b_j^* for the constants $a_i (i = 0, 1, \ldots, 5)$ and $b_j (j = 0, 1, 2)$ can now be determined by executing the synthetic job with different values for the parameters and measuring the CPU time used as well as the number of words transferred on the UNIVAC 494 system. We made 62 experiments to determine the estimators a_i^* and 65 experiments for the constants b_j^*, i.e. about 20% of N'_{tot}. The test conditions were identical to those of the total jobstream—that is to say normal conditions as they are found on our system every day. The parameters for the synthetic job were varied in an iterative

and empirical way between the following limits:

NPAS 1- 10
NMAS 0- 200
NDET..... 0- 100
NREP..... 0-10000
NPRINT ... 0- 330

resulting in a distribution of the experiments in the X_1-X_2-plane summing all intervals of the central memory—similar to that of the synthetic stream to be built (**Fig. 7**).

The values for the CPU time used and the number of words transferred, which resulted from these tests and which are known from the system logfiles, were used—together with the parameter values giving these resource requirements—for determining the estimators a_i^* and b_j^* by means of a linear regression analysis.

Using the following abbreviations

NPAS ..... u_1
NMAS + NDET ..... u_2
NPAS.(NMAS + NDET) ..... u_3
NDET.NREP.NPAS ..... u_4
NPAS.NPRINT ..... u_5

equation (7′) can be written as follows:

$$y_i = a_0 + a_1.u_{1i} + a_2.u_{2i} + a_3.u_{3i} + a_4.u_{4i} + a_5.u_{5i} \quad (9)$$

where $i = 1, 2, \ldots, 62$ denotes the index of the experiment. u_{ki} provides the values of the selected parameters of the synthetic job and y_i the measured CPU time used according to these parameter values. The equation (8′) for the number of transferred words can be rewritten in a similar form. Generally speaking a linear model is written as follows:

$$\vec{y} = U.\vec{a} + \vec{e} \quad (10)$$

where $\vec{y}$ is a $(n \times 1)$ vector of observations, U an $(n \times k)$ matrix of known coefficients $(n \geqslant k)$, $\vec{a}$, a $(k \times 1)$ vector of parameters and $\vec{e}$ is an $(n \times 1)$ vector of 'error' random variables with $E(\vec{e}) = 0$ and dispersion matrix

$$V(\vec{e}) = E(\vec{e}.\vec{e^T}) = \sigma^2.I$$

where I is the $(n \times n)$ identity matrix.

The assumption that the model is a linear one can be justified by Fisher's test (Kendall *et al.*, 1973). The least square method to solve the problem of estimating the parameters $\vec{a}$ is well known. Since the conditions for a full rank linear model exist the estimators together with their confidence intervals can be determined in an unequivocal way (see below).

Therefore equations (7′) and (8′) were written as follows:

$$\text{CPTIME} = 0{\cdot}00304 + 0{\cdot}08518\ \text{NPAS} + 0{\cdot}00509\ (\text{NMAS} + \text{NDET}) + 0{\cdot}00820\ \text{NPAS}.(\text{NMAS} + \text{NDET}) + 0{\cdot}00054\ \text{NDET.NREP.NPAS} + 0{\cdot}01296\ \text{NPAS.NPRINT} \quad (11)$$

$$\text{TRANS} = 16976{\cdot}61 + 1{\cdot}00572\,(2.\text{NPAS} + 1).(\text{NMAS} + \text{NDET} + 2).265 + 28{\cdot}22512\ \text{NPAS.NPRINT} \quad (12)$$

8. Construction of the synthetic jobstream

The next step was to determine the values of the characterising parameters of the synthetic jobs in such a way that the resulting requirements corresponded to the mean values of the real workload in each cell, which resulted from equation (3) and were described in Section 5. That means that for the given values of CPTIME and TRANS the parameters NPAS, NMAS, NDET, NREP and NPRINT had to be determined according to the regression equations. For this purpose equations (11) and (12) were transformed into

$$\text{NREP} = (\text{CPTIME} - 0{\cdot}00304 - 0{\cdot}08518\ \text{NPAS} - 0{\cdot}00509.(\text{NMAS} + \text{NDET}) - 0{\cdot}00820\ \text{NPAS}.(\text{NMAS} + \text{NDET}) - 0{\cdot}01296\ \text{NPAS.NPRINT})/(0{\cdot}00054\ \text{NDET.NPAS}) \quad (13)$$

$$\text{NMAS} + \text{NDET} = (\text{TRANS} - 16976{\cdot}61 - 28{\cdot}22512\ \text{NPAS.NPRINT})/(1{\cdot}00572\,(2.\text{NPAS} + 1).265) - 2 \quad (14)$$

The problem was solved by the procedure described below. From equation (14) NMAS + NDET was calculated with NPAS = 1 and the known value of NPRINT. This sum must in any case be split into two values, i.e. one for NMAS and the other for NDET, always taking into account that NMAS must be larger than or equal to NDET. In our case this subdivision was made according to the ratio between the number of disc transfers and the number of drum transfers. By means of these values for NMAS and NDET NREP was calculated from equation (13). When NMAS or NDET became too large, (>300) or (>400,000) respectively, the whole procedure was repeated using the value of NPAS incremented by 1. This limitation of NMAS and NDET was necessary to prevent the reservation of too large an area of secondary storage. The limitation of NREP was to prevent an integer overflow. To be sure that rounding errors with this procedure were acceptable, the values for the calculated parameters were substituted in equations (11) and (12). The resulting values for CPTIME and TRANS were later compared with the given ones. When the deviation was larger than a given tolerance the procedure was repeated with a new value of NPAS by incrementing the old one by 1.

a		*confidence interval*	
index	*value*	*lower limit*	*upper limit*
0	0·00304	−0·93609	0·94217
1	0·08518	−0·17332	0·34367
2	0·00509	−0·00139	0·01157
3	0·00820	0·00550	0·01090
4	0·00054	0·00054	0·00055
5	0·01296	0·00897	0·01712

b		*confidence interval*	
index	*value*	*lower limit*	*upper limit*
0	16976·61004	13487·79386	20465·42623
1	1·00527	0·98951	1·02103
2	28·22512	22·14894	34·30130

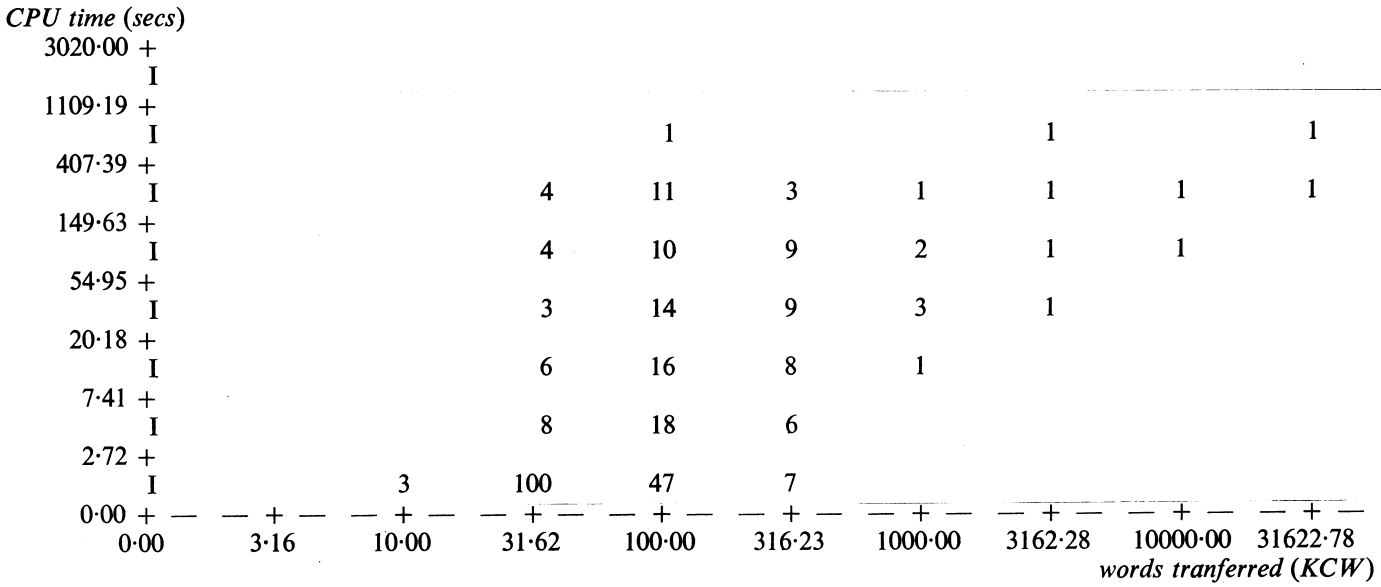

Fig. 7 Distribution of all synthetic jobs (summing over all memory intervals)

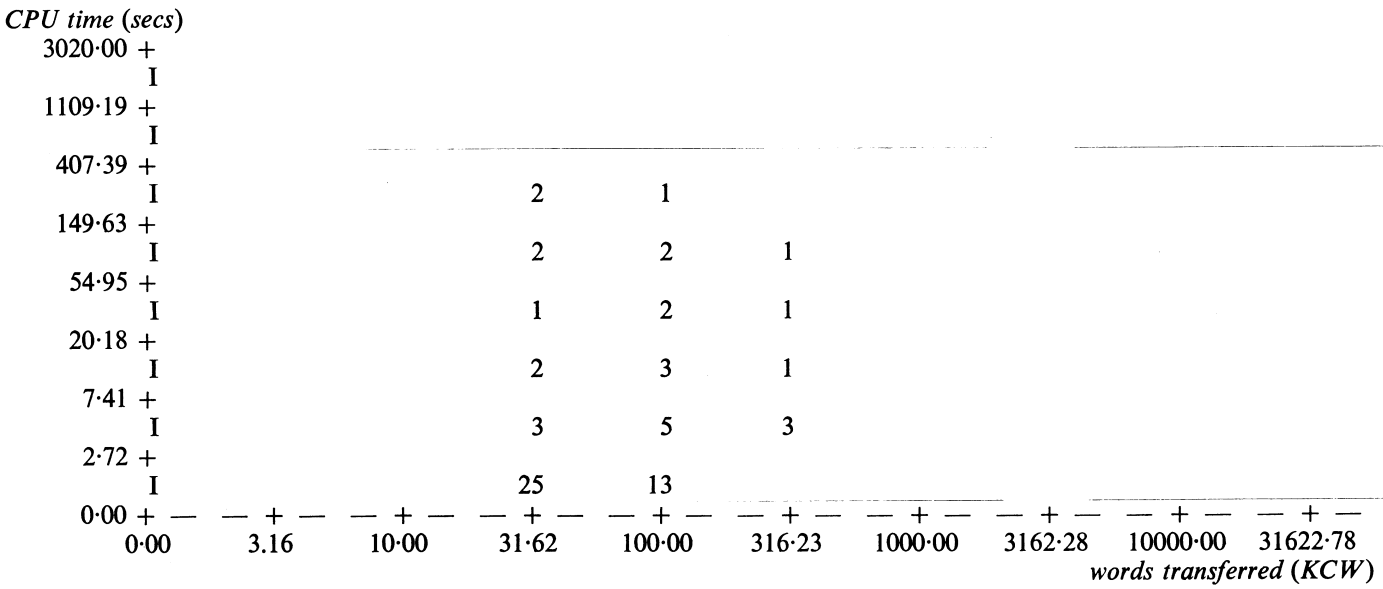

Fig. 8 'Real' distribution of the synthetic jobs in the core interval 4-8 KCW

With this procedure the parameters of all N'_{tot} jobs were determined.

Fig. 8 shows the 'real' distribution of the synthetic jobs in the core interval 4-8 KCW, after this procedure. The differences from Fig. 2 are: one job in the CPU time interval (0·00, 2·72) moved from the interval (10·00, 31·62) for transferred words into the next interval (31·62, 100·00) because the transfer requirements of this job were near the margin of these two intervals; two other jobs which were within the transfer interval (100·00, 316·23) shifted up from the CPU time interval (0·00, 2·72) to the next interval (2·72, 7·41). This was due to the fact that both jobs had to print many lines which could not be done within the given CPU time. These were jobs where the execution phases result from assembler input printing lines faster than can be done in FORTRAN (fast print jobs). Finally the sequence of jobs in the synthetic stream was fixed by random selection.

9. Conclusion and experiences

This paper described all the steps necessary to construct a synthetic jobstream, the resource requirements of which match those of the real batch workload running on a UNIVAC 494 system in a university environment. After completion of these test runs on three different systems we can say that for the selection process this synthetic jobstream was useful with the following advantages:

1. It is a practicable method of constructing a realistic test run which can easily be adapted by vendors of a system to be tested.
2. After completion of the theoretical work it is a quick way to construct a valid representation of a given batch workload assuming that enough information on this load is available.
3. According to our experience we are able to say that the results of this method can not be replaced by others, for instance by executing sample jobs or by theoretical considerations.

It should be mentioned that it is necessary to give an exact procedure for the execution of the test run to obtain comparable results (e.g. a better optimisation of the compilation of some of the jobs can only be allowed if required for all).

The restriction to a pure batch load was necessary because the existing system had no real TS facility. The resulting synthetic batch load—together with a synthetically predefined and simulated TS load based on assumptions as far as the qualitative and quantitative nature of the most probable

work to be done on a terminal in a university is concerned—defined the total test run to be executed on the systems to be compared in a selection evaluation process, resulting in the replacement of the existing system by a new one.

References

BUCHHOLZ, W. (1969). A Synthetic Job for Measuring System Performance, *IBM Sys. J.*, Vol. 8, No. 4, pp. 304-318.

GELL, G. *et al.* (1979). Entscheidungskriterien und Ergebnisse des Auswahlverfahrens bei der Planung eines Universitätsrechenzentrums to be published.

HUNT, E. *et al.* (1971). Who are the Users? An Analysis of Computer Use in a University Computer Center, *AFIPS Conf. Proc.*, Vol. 38, pp. 231-238.

KENDALL, M. G. *et al.* (1973). *The Advanced Theory of Statistics*, Vol. 2, Ch. Griffin, London.

KLEINROCK, L. (1976). *Queueing Systems*, Vol. 2, Computer Applications, J. Wiley, New York.

LUCAS, H. C. (1971). Performance Evaluation and Monitoring, *ACM Comp. Surv*, Vol. 3, pp. 79-91.

OSSWALD, B. (1973). *Leistungsvermögensanalyse von Datenverarbeitungsanlagen*, S. Toeche-Mittler, Darmstadt.

POSCH, R. *et al*, (1979). Ein leistungsorientiertes Bewertungsverfahren zur Auswahl von Großrechnern, to be published.

SREENIVASAN, K., and KLEINMAN, A. J. (1974). On the Construction of a Representative Synthetic Workload, *CACM*, Vol. 17, pp. 127-133.

Book reviews

Distributed Processing and Data Communications, by D. R. McGlynn, 1978; 305 pages. (*Wiley*, £14·50)

There is a computer somewhere which has a splendid piece of software running in it. The suite is known to the cognoscenti as AFABOG, or Automatic FAshionable BOok Generator. It runs in a background partition, monitoring trendy buzz terms as they flash through the trade press. As soon as more than five conferences/seminars with a particular buzz phrase in their promotional handouts have been held, instructions are sent to a publisher somewhere to commission a book with the appropriate buzz terms in the title. The suite is quite clever. It extracts subheadings from conference handouts and correlates them with previously successful chapters in books and articles in the trade press. From this a picking list is generated which is sent to the author. All that the human being has to do is pad out the paragraphs to make up chapters. The system does the rest—it sells the product, despatches, invoices, deals with complaints and pays royalties to the author.

This book is a splendid example of the work of AFABOG. It has caught the 'distributed processing' and 'data communications' markets slap on the crest of their individual fashionability waves. A couple of years ago it was all 'structured programming' and 'data base management'—next year it will be 'microprocessing' and 'office automation'. The contents of the book are not particularly interesting. It is basically a cull of previously published material, reduced to its elementary level, and slightly repackaged. It is difficult to see just what market the book is aimed at. On one level it is full of technical guff like 'Modified Frequency Modulation has a recorded bit density . . . of 6536 frpi, compared to 8170 frpi for GCR code. On the other hand MFM has a phase margin of 0·125 bit cell time compared with 0·20 for GCR'. On another, it is supremely lacking—in the section on data comms software it discusses IBM systems software products only, and unless the reader is already familiar with IBM's marketing practice, he is left very little wiser and probably very confused.

One of the fundamental problems of distributed processing continues to be that of distributed data base, and maintaining the integrity of it. This book ducks the question head-on. In four paragraphs a tiny overview is given, and dismissed—'The topic of distributed data bases is a complex one that can only be briefly outlined in this work'.

There is a chapter on network control architectures. This is most illuminating, not for its technical content, but more for the insight it gives the reader into the author's own biases. For example, X25/HDLC earn 1½ pages, Burroughs data link control gets 3 pages, and SNA/SDLC gets 18 pages; Decnet earns 14 pages. There is no attempt made to compare and contrast the eight control architectures, and the reader is left with the impression that the only worthwhile approaches are those of IBM and DEC. While this might be true, it certainly is not a professional thing to do, to make the uninformed reader believe it without going into the pros and cons of the argument.

The most useful part of this publication is the back end. There is a good index, a fair glossary and a very useful appendix containing a systems man's checklist on the considerations of implementing data comms. Called 'Technology forecasting and assessment', it is a quite rigorous walkthrough of the salient points. If the book retailed at a third of its price, it would be worth buying for this last part. As it is . . .

SEAN O'CONNELL (Woodley)

Quick Cobol, 2nd Edition, by L. Coddington, 1978; 257 pages. (*Macdonald and Jane's*, £6·50)

The aim of Mr Coddington's book is twofold. Firstly and primarily to give users of computers a knowledge of the most commonly used business computer language; secondly to give computer personnel an easier introduction to COBOL.

The first aim succeeds to a greater extent. The book is easily readable and the only question must be whether any user of computers needs such extensive knowledge of a computer language. The success of the book as far as the second group is concerned is more questionable. Although the book does give a good introduction to COBOL there have been considerable changes in methods of programming since the first edition of the book in 1971. Mr Coddington discusses these in a fairly perfunctory manner recommending the reader to a book on advanced COBOL for more detailed information. This seems to suggest that alterations to COBOL have all been in the more advanced use of the languages whereas, for example, the new indexing methods are probably easier to understand to someone learning the language. It is also slightly disappointing to find structured programming covered in one paragraph in the preface.

I have not read the first edition of the book but would have thought that in a second edition more changes could and should have been included.

J. EMERSON (Horsham)

Specimen copies

Information and Management is the Journal of the IFIP Users Group (IAG) published by North-Holland. Volume 2 number 1, February 1979 contains an article by Isaac Auerbach, a founder and first President of IFIP. He is a Distinguished Fellow of the BCS, among his many honorary titles.

The guest editorial by Howard Resnikoff covers the need for research in information science.

Free specimen copies of this publication are available for qualified readers of the Journal on application to

Mr J Dirkmaat
North-Holland Publishing Co
PO Box 211
1000 AE Amsterdam
The Netherlands