Performance evaluation of a process control system by simulation model*

M. Boari, P. Pellizzardi, and R. Rossi

Istituto di Automatica, Facoltà di Ingegneria, Universitàdegli Studi di Bologna, Viale Risorgimento 2, 40136 Bologna, Italy

In this paper the problems met with in the performance evaluation of a process computer system are faced. After pointing out the meaning of performance evaluation for such an application and identifying the set of measurements characterising the behaviour of the controlled physical process and the computer system, the evaluation method employed is described; this consists of a set of direct software measurements and a simulation model fed by event trace.

The simulation model structure is analysed and some problems relating to its feeding are illustrated. To facilitate choice of the set of measurements to be carried out and their interpretation, the system behaviour is described through a model derived from the Petri net. Finally, the results relative to the behaviour of a computer system dedicated to the control of a gaschromatographic laboratory is reported.

(Received January 1973)

1. Introduction

In studying the performance evaluation of time-sharing systems, which have up to now attracted the greatest interest in this field of study, the basic measurements typical of its behaviour are: throughput, turnaround time, availability (Lucas, 1971). In this kind of work the response to the user must be given in a reasonably short time with, besides, a good resource utilisation and a fixed degree of availability.

This kind of environment (Manacher, 1967) is characterised by soft real time requests. In fact, response time is not the only characteristic of the system behaviour, thus small variations in the service may, in general, be tolerated provided they permit better utilisation of the resources.

In a computing system for process control, the world external to it, the process, usually presents the computer with a set of rigid timing bounds. In fact, by calling $t_1(j)$ the time in which the request j is sent to the computer, and $t_2(j)$ the resulting deadline of the service request, the following condition must be true:

$$t_2(j) - t_1(j) \geq t_{el}(j) \;;$$

where $t_{el}(j)$ is the maximum execution time of the application program j, depending not only on its own characteristics but also on the working environment, that is, the computer system configuration, the number and type of programs, the frequency of request and, finally, the resource allocation policy.

The basic aim of the performance evaluation in this case consists in verifying whether, and to what extent, (1) is satisfied, and in pointing out possible causes for ignoring the timing bounds.

The set of measurements to be carried out must permit analysis of the dynamics of the interactions between the computer and the process. In such a case the increase in throughput becomes of secondary importance, presenting itself only if the computer system is predisposed for off-line work during the time intervals where process requirements do not exist. In fact, keeping within the timing bounds it is, in this case, possible to organise a policy of resource allocation so that the system performances increase.

2. Evaluation method

The choice of evaluation method depends on the aims to be achieved by means of performance evaluation and, consequently, on the kind of measurements to be carried out.

In a recent work (Noetzel, 1971) the utility has been shown of an evaluation method jointly employing a direct software measurement technique and a simulation model fed with the data obtained from the measurements. (**Fig. 1**).

The use of direct measurement techniques does, in fact, lead to detailed information on the system behaviour sufficient for performance evaluation (Boari, Neri and Pellizzardi, 1972). When, however, by employing the results obtained, it is desired to verify the consequences on system performance of modifications in its configuration, both hardware and software, a simulation model is advisable. Such a model must be fed by data obtained from the event trace relative to the application programs resource request.

These data must be independent of the hardware and software features of the system whose influence on the performance is investigated by the simulation model; in fact, only in such a case do the data maintain their validity even if modifications to the original organisation of the system have been introduced into the model.

If the system is organised so that a virtual environment is created for each application program, requests for system resource utilisation are clearly independent of the procedure by means of which the operating system satisfies them. When, however, this type of organisation is not available, as in the system described in this paper, the events obtained relative to the application program behaviour are not independent on the system characteristics. To achieve this independence, the event trace must be transformed.

The evaluation method adopted allows satisfactory solution of the simulation model validity verification problem. In fact, it is possible to compare the results obtained from the direct measurements with those from the simulation model reproducing the original configuration of the system.

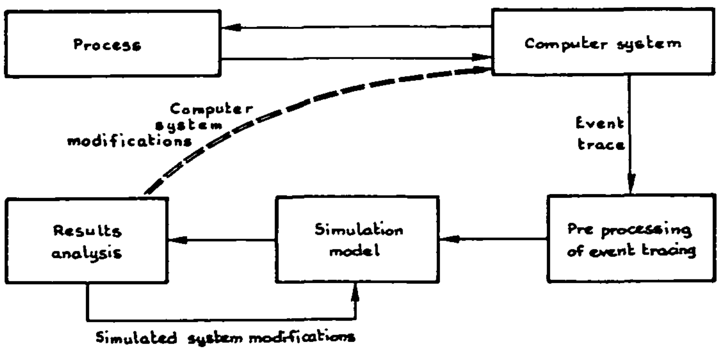
Process
Computer system
Computer system modifications
Event trace
Results analysis
Simulation model
Pre processing of event tracing
Simulated system modifications

Fig. 1 The steps of evaluation method

*This work was supported by CNR (Consiglio Nazionale delle Ricerche) under grant number 70.02149.07 and by ENI (Ente Nazionale Idrocarburi) under grant numbers 1834 and 1992.

3. Description of simulation model

The computer used is a GE 4020 dedicated to the control of gaschromatographic laboratory.

The hardware resources available in the present configuration consist of a CPU, a core memory of 24K words (each word is of 24 bits) with a cycle of 1·6 μ-seconds, and a moving head disc capable of one million words.

The computer runs under the RTMOS (Real Time Multiprogramming Operating System), the resident part of which (resource allocator, scheduler, drivers) occupies about 7K words of the core memory. The remainder of the operating system is disc resident and consists of:

(*a*) the logical I/O represented in the system as a set of high-priority programs;
(*b*) a small conversational system;
(*c*) the compiling system and library management at low priority.

The computer addressing technique permits an elementary form of dynamic allocation by using the program counter as relocation register. The programs are allocated as consecutive memory addresses and the choice of memory areas to assign them is made in accordance with the best fit technique. In the present system the possibilities of this technique are not completely exploited since relocation and software organisation sometimes links a program to a particular memory area.

The simulation model realised represents in detail all the functions of the operating system relative to CPU scheduling, I/O scheduling and memory management. These functions are obtained by means of interacting subprograms, the execution of each of which, in accordance with the request priority for activation of the functions, determines simulated system evolution.

Such a structure, while allowing representation of parallel activities, confers considerable modularity to the model. Thus, the modification of the hardware configuration, the resource allocation techniques and the controlled process description, are facilitated.

The data necessary for a simulation run may be subdivided into three categories:

(*a*) hardware configuration data of the computer system: core memory size and cycle time; number of channels, I/O devices, auxiliary memory units as well as their access time and transmission speed characteristics;
(*b*) operating system data; execution time of the operating system modules obtained from direct measurements on the system and their priority in using CPU;
(*c*) application program data: description of their behaviour in terms of hardware and software resource system request.

As regards (*c*) it should be noted that two representation techniques are possible: the first one is based on the measured event trace (Cheng, 1969), the other is based on a behaviour representation of the generic program, where resource requests to the system are expressed by probability distribution.

If the possibilities offered by the above techniques are examined, it will be noted that the first one, allowing a more faithful description of workload behaviour, is particularly suitable when using the simulated model, if it is desired to test hardware or software system modifications, and workload modifications that do not alter program characteristics.

As it may happen in process control computer systems, when workload variation, besides modifying the program execution rate, alters their characteristics (e.g. execution time, number of I/O operations, etc.) it is necessary to employ statistical representations which, in fact, make program behaviour description possible as a function of the parameters representing the system workload.

On the basis of the above, since one of the aims of this paper is to determine the maximum workload to which the system may be subjected, it was decided to adopt a statistical representation.

For every program is given:

(*a*) execution rate;
(*b*) duration;
(*c*) execution time between two successive I/O operations;
(*d*) the number of seconds sent for each I/O operation;
(*e*) memory occupation;
(*f*) I/O probability on the channels;
(*g*) desired response time.

(*a*), (*b*), (*c*) and (*d*) are supplied by means of normal distribution whose mean value and variance are obtained by direct measurement on the system. (*e*) and (*f*) are given deterministically on the basis of the results obtained from the direct measurements, whole (*g*) is a planning datum.

The program description model affects only one simulator subprogram and can be changed without overmuch reprogramming.

The time resolution obtainable, of course, depends on the basic time unit assumed; in the model it is variable and is chosen in accordance with the time resolution power of the measuring technique which is of 250 μ-seconds. Generally, depending on the workload, it takes $5 \div 15$ μ-seconds of CDC 6600 to simulate a minute of GE 4020.

4. Definition of the set of measurements

As previously seen the basic aim in the performance evaluation of a process control system consists of verifying whether, and to what extent (1) is satisfied for a given workload, and in pointing out the possible causes which lead to ignoring of the timing bounds.

Satisfaction of condition (1) depends on the controlled process characteristics, that is, on the behaviour of the application programs and the dynamics of their interactions, as well as on system resource availability and the management policy adopted.

By analysing the ratio:

$$\mu = \frac{t_{el}(j)}{t_P(j)}$$

between real execution time $t_{el}(j)$ (measured from activation to the end of program j) and the time $t_p(j)$ (the sum of the times during which the program occupies each resource) it is possible to have some indication as to the extent of the delay in execution caused by resource competition between the programs.

Further information can be drawn from the level of hardware and software resource utilisation. Yet such information is insufficient if it is decided to investigate to what extent the response time is affected both by the process characteristics, and by the system hardware and software organisation. It is necessary, to this end, to have a model of the system which allows its work to be described with the desired accuracy, thus facilitating both the choice of the set measurements to be effected and the interpretation of their results.

This model, based on the computer system hardware organisation and the operating system structure, must indicate how program flow develops through the system, keeping possible parallelism of the operations in mind as well as the sequence and conditions in which the resources are occupied.

It has been shown (Noe, 1971) how a model based on a modified Petri net can be used to represent a highly parallel system and how this representation leads itself to measurement planning. Although the present does not belong to the high parallel system category there are two important features of the Petri net which are of considerable interest: the representation of concurrent events and the easy modification of the level of detail being displayed.

The elements of a Petri net are: the conditions existing in a

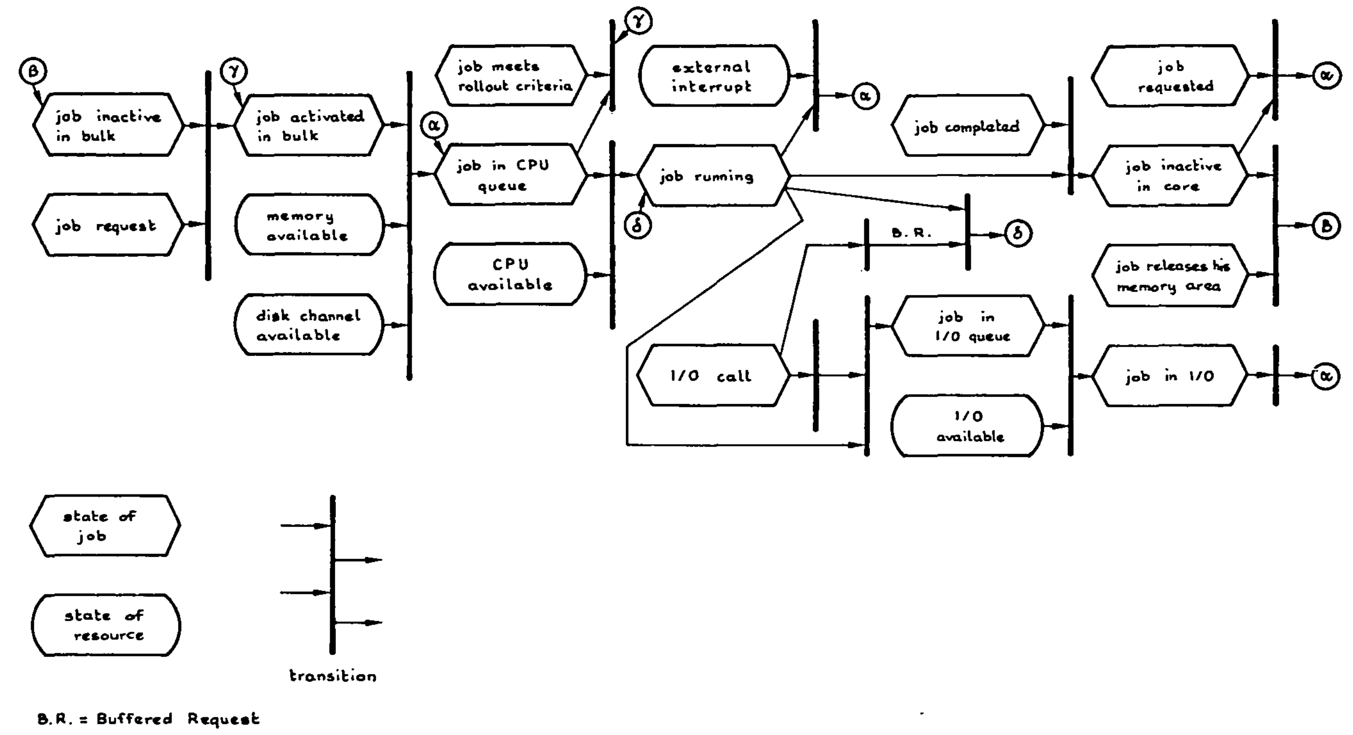

Fig. 2

certain time interval and the events (transitions) which, determined by the existence of a set of conditions, cause the passage to another set of conditions. In the present case the conditions show that the programs and resources remain in particular states, while the transitions indicate the passage from one set of states to another.

A transition takes place when all the necessary logical conditions are simultaneously satisfied. Two or more parallel conditions between two transitions are considered concurrent even if a more detailed representation of the working system shows them to be differently placed in time.

For the present system we have obtained the model in **Fig. 2.**

On examining the Petri net it is clear that, by measuring the number of times a transition takes place and the time interval between two successive transitions, one can evaluate the duration of the stay of each application program in a particular state and the frequency with which the necessary logical conditions take place to originate a transition. It is possible through these measurements to learn how the flow of each program through the various states is conditioned by its characteristics and the state of the system. In particular, it is therefore possible to know the extent to which the non-availability of resources is responsible for the stay of each program in some states, thus modifying the response time.

5. Result analysis

Verification of the hypotheses on which definition of the simulation model is based, has been obtained by means of comparison, in the same workload conditions, of the results obtained from the model itself with those measured directly on the real system. These results, limited to the response times of each application program and the extent of resource utilisation, are given in **Table 1** and show satisfactory agreement.

During the measuring phase, the off-line workload on the system engaged it completely. The value obtained for the off-line calculations thus represents the maximum work the system can do compatible with the load imposed by the process.

One of the aims to be reached by the simulation model is the maximum value of the process workload that the computer system can tolerate without exceeding the response time bounds imposed by the process on some application programs. Besides, at such value, it is desired to verify the quantity of off-line work that the system can still develop.

In the particular process controlled, the workload consists of a set of different gaschromatographical analyses.

Workload variations are realised by acting on the number of analyses carried out in a unit time keeping constant the proportions between different kinds of analyses.

The gaschromatographical package operates as follows: the gaschromatographical data are stored directly on the disc by the analogue input drivers: every 24 seconds an application program (program 2) orders and packs them in the files related to each gaschromatographical.

At the end of the analysis the identification of the chemical components is performed together with the relative quantitative determinations (program 4). The results of this step are sent to programs 5 and 6 which carry out further calculations and print the analysis report.

While the constraint on program 4 response time is due to the filling up of the disc memory available for the data to be worked out and must be not more than a few minutes, programs 5 and 6 have no rigid bounds imposed by the process by the system dynamic but only those connected to the laboratory needs which requiie a response time of about 30 minutes.

During these operations program 1 checks the correct working of the gaschromatograph every two seconds while program 3 time constraints closely depend on the 'control procedures' used with the same instruments; the actions carried out by this program must not be delayed for more than 10 seconds.

All the programs are disc resident and their length varies from 4–5K for programs 4 and 6 while the others have lengths between 1–3K.

Fig. 3 shows the results obtained by varying workloads, for the response time of the application programs which are more critical to time bounds.

Measures based on Petri's net give for each program the average time spent in each state and the frequency at which they are engaged.

From an analysis of these results we can know which states affect the real execution time of the programs in a decisive manner.

Results relative to such measures, shown in **Table 2**, underline a high permanence time for the central memory waiting state, essentially due to the non-availability of one or both resources, memory and channel.

As the waiting time in the I/O queue state (Fig. 2), essentially determined by channel waiting-time, is within acceptable limits, the hypothesis may be advanced that the bounding resource is the memory which, on the other hand, is employed to an extent greatly inferior to its capacity.

It is therefore reasonable to suppose that the cause of excessive program permanence in the central memory waiting state, is to be found in the memory management technique adopted in the computer system.

To verify this hypothesis in the simulation program a form of management has been realised allowing, when necessary, compacting of the memory areas occupied by the programs.

The managing technique adopted produces a remarkable reduction in waiting time of the considered state (see Table 2);

Table 1a Execution times

	Real system	*Simulated system*
	(seconds)	(seconds)
Program 1	0·53	0·54
,, 2	1·82	1·89
,, 3	3·26	3·42
,, 4	9·45	8·78
,, 5	0·87	0·83
,, 6	19·40	21·18

Table 1b Resources utilisation

	Real system	*Simulated system*
	(per cent.)	(per cent.)
CPU:		
Application programs	10·5	9·9
Programs off-line	82·4	80·7
Overhead	4·6	5·2
Idle time	2·5	4·2
Central memory	33·4	34·2
Disc channel	20·3	20·7

Table 2 Average time of the programs in the various states

	Without compacting		*With compacting*	
	Average time (ms)	*Relative frequencies*	*Average time* (ms)	*Relative frequencies*
Job activated in bulk	1560	0·295	985	0·26
Job in CPU queue	19·7	12·0	21·4	11·0
Job running	30·2	12·0	34·6	11·0
Job in I/O queue	208	4·2	163	4·2
Job in I/O	93·6	4·2	86	4·2

The results are relative to a workload of 3·95 analyses/minute.

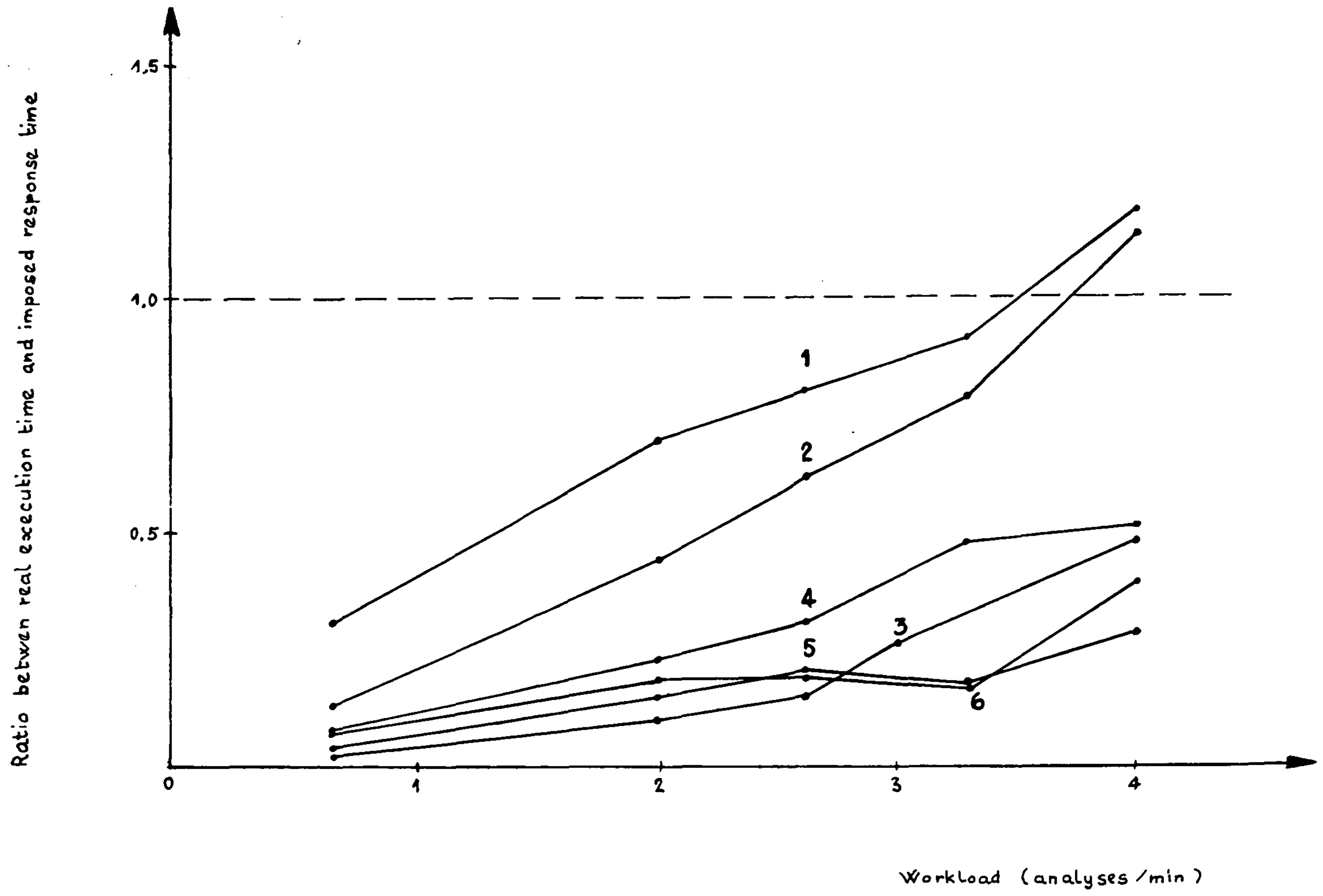

Fig. 3

Table 3 **Comparison of resource utilisation**

	Without compacting and fixed priority	*With compacting and fixed priority*	*Without compacting and 'least time to go'*	*With compacting and 'least time to go'*
CPU utilisation:				
application programs	18·5%	18·1%	18·3%	18·1%
off-line programs	47·5%	59·4%	46·9%	53·0%
overhead	29·1%	15·1%	29·4%	12·8%
idle time	4·9%	7·4%	5·4%	16·1%
CPU average waiting time:	19·7 ms	21·4 ms	20·1 ms	19·1 ms
Memory utilisation:				
application programs	51·4%	48·2%	53·4%	50·0%
off-line	16·0%	16·0%	16·0%	16·0%
Memory average waiting time	1560 ms	985 ms	1341 ms	987 ms
Disc channel utilisation:				
application programs	54·0%	58·7%	54·8%	57·0%
off-line programs	2·9%	3·7%	2·8%	3·2%
overhead	11·2%	6·8%	10·9%	7·0%
Disc channel average time	141 ms	133 ms	127 ms	169 ms
Maximum time spent in channel queue	887 ms	897 ms	9260 ms	28620 ms

The results are relative to a workload of 3·95 analyses/minute.

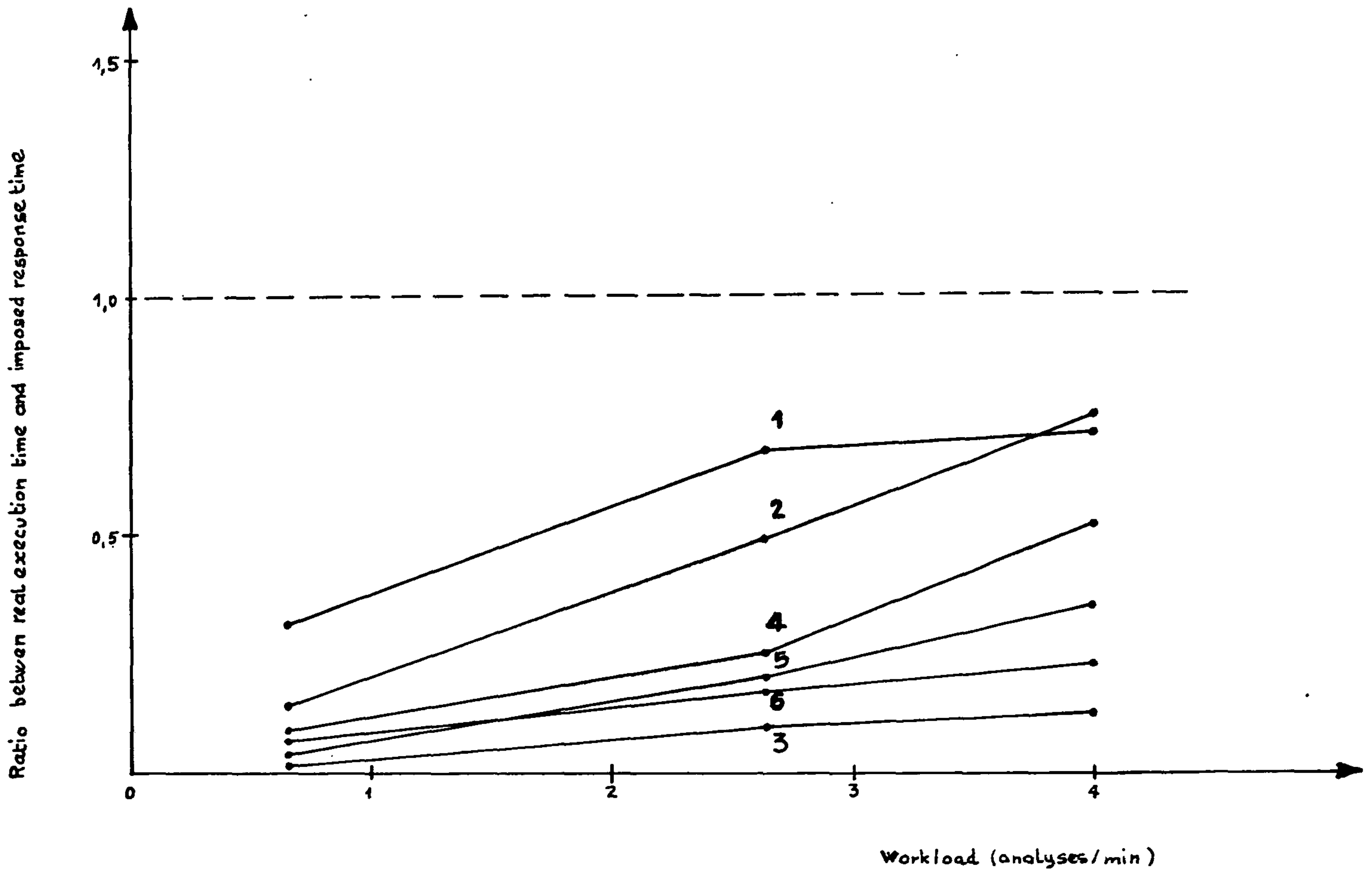

Fig. 4

however the low frequency at which programs engage the state limits the effect on their real time of execution.

On the other hand, results reported in **Table 3** show that the resource exploitation is considerably improved. This is essentially due to the reduced number of program transfers between the central memory and the disc, ensuing from better memory utilisation.

Therefore, the system can, with the same process workload, effect off-line work 25 per cent higher than before.

The notable reduction of the overhead obtained by compacting the memory can be justified by the few unsuccesful attempts of the operating system to allocate in the core the non-resident programs. This improvement however is in part reduced by the longest period of time in which there are no programs to be executed in the central processing unit.

These results and the high degree of overall resource exploitation obtained, make it reasonable to consider of limited efficacy any further attempts to improve program time performance on the basis of improved resource exploitation.

In the present system CPU and I/O management are realised, respectively, by fixed priority assignation and by a first-in first-out technique. Since program criticalness depends both on the state of the system and on the behaviour exhibited in the runs, it may be thought that an improvement in system time performance is possible by means of a management technique dynamically assigning priority in the use of CPU and I/O.

To this end a 'least time to go' technique has been employed which, as is known (Fineberg, 1967), assigns instant by instant maximum priority to the program with the most critical time bound.

The results obtained, shown in **Fig. 4** confirm the validity of the hypothesis; the response times for critical programs, remarkably reduced, are now within limits imposed by the process.

Naturally this improvement brings an increase in the response times of the non-critical programs, which are not discussed in this paper, and a lower level of resource utilisation.

A further investigation has been carried out to examine the effects due to the introduction of the 'least time to go' algorithm alone leaving unchanged the memory allocation policy.

The results relative to the utilisation of the resource are reported in Table 3 and show again the trend in increasing the overhead and decreasing the time available for off-line work.

However, in this situation the introduction of the 'least time to go' algorithm has not brought any noteworthy variations in the response times which substantially remain the same as those in Fig. 3 unlike what happened with the memory compaction policy.

References

BOARI, M., NERI, G., and PELLIZZARDI, P. (1972). Performance evaluation of process control systems, *XII Convegno internazionale dell' automazione e strumentazione*, 1972.

CHENG, P. S. (1969). Trace driven system modelling, *IBM System J.*, Vol. 8, No. 4, pp. 280-289, 1969.

FINEBERG, M. S., and SERLIN, O. (1967). Multiprogramming for hybrid computation, *Proc. AFIPS*, FJCC 1967.

LUCAS, H. C. (1971). Performance evaluation and monitoring, *ACM Computing Surveys*, Vol. 3, No. 3, September 1971.

MANACHER, G. K. (1967). Production and stabilization of real-time task schedules, *JACM*, Vol. 14, No. 3, July 1967.

NOE, J. D. (1971). *A Petri net model of the CDC 6400 – Workshop on system performance evaluation*, Harvard University, April 1971.

NOETZEL, A. S. (1971). The design of a meta-system, *SJCC*.

Book reviews

Topics in Numerical Analysis, edited by J. J. H. Miller, 1974; 348 pages. (*Published for the Royal Irish Academy by Academic Press*, £7·00.)

This text is sub-titled 'Proceedings of the Royal Irish Academy Conference on Numerical Analysis, 1972'. Of the nineteen papers published in the text, sixteen were given as one-hour invited papers at the conference. The remaining three are contributed by workers who were invited to speak but were unable to attend. Two invited papers are *not* included and are to appear elsewhere. These are 'Schwarz Functions and Iteration Theory' by P. J. Davis and 'The Hypercircle Method' by J. L. Synge. All papers are in English with the exception of that in French by R. Glowinski. This, at fifty pages, is also the longest.

Those who attended the conference were fortunate to hear two talks by Cornelius Lanczos. In addition to his invited paper 'Legendre versus Chebyshev polynomials' he gave a delightful evening lecture of a non-technical nature entitled 'Computing through the ages'. The full text of this excellent talk is presented as an introduction to this volume of research papers.

The overall standard of this collection of papers is high. They are arranged by alphabetical order of authors. As usual, partial differential equations are well to the fore, with papers by L. Collatz, E. G. D'jakonov, E. Schechter, R. Glowinski, P. A. Raviart, V. Thomée. The latter three papers are concerned with the finite element method. Ordinary differential equations are also well represented, with papers by R. K. Brayton and C. C. Conley, J. C. Butcher, J. Douglas and T. Dupont (a Galerkin approach, using splines), L. Fox *et al.*, H. O. Kreiss, H. Stetter. The latter paper is on discretisation theory, as is the paper which follows it, by F. Stummel. There are two papers on numerical linear algebra: one is by R. S. Varga with applications to the finite element method; the other is an attractive paper by G. H. Golub concerning the Lanczos algorithm for finding eigenvalues. There remain three papers: one on quadrature formulae by G. Freud, one on control theory by J. L. Lions and one on Fredholm integral equations, a notable paper by B. Noble.

The editor has done his job well, although some would wish to have seen more material on linear algebra and rather more than the solitary contribution (from Lanczos) on approximation theory.

G. M. PHILLIPS (St. Andrews)

The Skyline of Information Processing. Proceedings of the Tenth Anniversary Celebration of IFIP, edited by H. Zemanek, 1972; 146 pages. (*North Holland Publishing Co.*, $7.00.)

In Amsterdam in October 1970, IFIP celebrated its tenth anniversary by means of a set of eight lectures given by speakers, most of whom had been prominent members of the IFIP organisation during its formative years. This book records those lectures in print.

The lectures fall primarily into two categories, historical and philosophical. In the first category fall topics such as 'Ten years of IFIP' and 'Computers and Technology', into the second come 'Need for an information systems theory' and 'Some philosophical aspects of information processing', while one paper 'IFIP and the expanding world of computers' falls neatly into both camps.

The majority of lecturers are from non-English-speaking countries, yet the style is uniform and the use of English is immaculate; one must assume that an excellent job of editing has been done. The style also is one that makes the entire book very easy to read. Nevertheless, it is difficult to see who would want to buy the book or where it would fit on one's bookshelves. The historical information is too lacking in detail either to be useful as a definitive account of the early history of IFIP or to provide a textbook in any aspect of computer science. The philosophical offerings again, are too generalised to show the direction in which one should direct one's efforts in the future, while the information content about current practice is not such that any on-going project is likely to be changed as a consequence. Although the sets of references given at the end of two of the lectures are comprehensive, many better ones have appeared before.

Naturally, if one wishes to have a printed record of a historic event, or to refer to the current thoughts of some of our most distinguished colleagues, or even to be given some hints on how to begin a new approach to computing's problems, then here is a purchase well worth making. I fear, however, that the potential market in this area is likely to be small.

Some of the papers could serve as introductory reading for executive appreciation although here, again, the cover is not sufficiently wide to be valuable.

P. HAMMERSLEY (Cambridge)