Tuning to service standards without monitors

J. F. A. Wiederhold

Computer Centre, University of the Witwatersrand, 1 Jan Smuts Avenue, Johannesburg 2001, South Africa

A technique, based on an analytical model, a benchmark job stream and information routinely provided by an operating system, has been developed to assist computer centre management with hardware selection and system tuning. In this paper the author proposes the extension of the technique of system tuning in relation to service standards.

The technique requires neither additional equipment nor specialist personnel so that most installations can implement the method with relatively little expense. The use of information provided by the operating system means that evaluation is described in terms of the conditions which prevail during production. The results constitute a satisfactory first pass at system evaluation and highlight areas that will need further investigation with hardware or software monitors.

Results include service/cost relationships, since a component of the model is the representation of service standards. Case studies are presented which illustrate the application of the technique in investigating optimum multiprogramming level and expected throughput capacity on different configurations.

(Received July 1976)

Computer system evaluation

Evaluation of computer systems is usually conducted for the purpose of selecting and configuring a new computer system, improving the performance of an existing installation, maintaining desired performance standards or predicting effects of changes in either an existing system or the demand workload.

Performance measurement and evaluation, pertinent in any system evaluation, embraces aspects ranging from hardware and software to operating philosophies. The emphasis on specific aspects, for example modes of access to the computer, system overheads, scheduling techniques, configuration changes and speed of certain operations, is a matter of policy decision and a function of the type of system evaluation required.

Timmreck (1973) reporting on the literature of evaluation methods, defines the analysis steps as follows: 'There are four basic steps in the selection process: analysis and specification of need, request for proposals, validation of proposals including system performance measurement, and the actual system selection'. Each step varies in magnitude depending upon the particular evaluation being conducted.

The performance measurement and evaluation phase is obviously the principal component of a system evaluation process when the object is to maintain desired performance standards or predict the effects of changes in the operating procedures, workload or configuration. This component is generally of lesser significance when evaluation focuses on selection of a new computer system since then, provided the performance of the system complies with certain minimum standards, aspects such as system facilities, financial criteria (including cost effectiveness over a period of time) and contractual details are likely to be of greater importance. In this paper, however, attention is restricted to performance evaluation and measurement.

Evaluation constraints

Any evaluation technique for performance studies must satisfy a range of constraints, varying from cost to versatility, which, in any given practical application, are likely to be imposed to some degree or another. This technique, moreover, must provide results in a form readily useful to those who will be responsible for assessing their significance. The 'level of concern' as specified by Kimbleton (1973), that is to be addressed in this paper, is the performance of the computer operations environment. The performance of this 'level of concern' may be influenced by activities of 'inner systems'. As a result, 'inner systems' evaluations may have to be included.

The first constraint which must be imposed upon an evaluation technique that purports to be general, is that of portability. In other words, the technique must, as far as possible, be independent of supplier company's hardware and software. A particular advantage of a technique which satisfies this constraint is that it permits comparisons even when architecture of hardware, operating system philosophy, and so on, differ considerably.

A second constraint to be satisfied by an effective evaluation technique is that continuous recording of performance information be possible so that day-to-day production standards can be measured and controlled. It is vital, in fact, that a common yardstick be used for both day-to-day production evaluations and those required by planning.

Finally, the cost of obtaining the necessary data must be kept at a minimum. This constraint almost precludes software and hardware monitors as the latter are expensive and the former significantly reduce productivity, and thus increase costs, when used on a continuing basis.

The above constraints have been aptly summarised by Schumacher (1973) who suggested that management's expectations of a performance evaluation effort is characterised as

(*a*) to define computer system performance in simple and relevant terms

(*b*) to establish the computer system's level of performance

(*c*) to establish methods to control computer system cost/performance.

Performance criteria

Service standards

It is common practice to express evaluation results in terms of utilisation factors. This information is, however, meaningful only within a relatively limited sphere since neither management nor users ordinarily associate such measures with their experience. As a result, the latter group cannot fully perceive and consequently interpret the information so presented. Such a situation is hardly tolerable since management needs a ready understanding of the concepts employed in defining system performance, to know what performance level is being achieved as compared with that which can be achieved and to be assured that controls exist which permit attainment of

minimum computer system cost/performance levels.

A further difficulty which is encountered when evaluation results are presented in terms of utilisation factors is that it is difficult to use such data to compare various computer configurations. A 60% utilisation of one resource might in fact lead to a 20% utilisation of another. What conclusion must therefore be drawn from a knowledge of such factors? Although this example illustrates the difficulty of interpreting the significance of utilisation factors and indicates that in isolation such data may be of limited value. Utilisation, as will be shown, is nevertheless an essential part of a more universal measure that exists in all production environments.

A more meaningful approach than that of utilisation factors *per se*, when measures are sought to express results of a performance evaluation of a computer system in a quantitative manner, is to present such information in terms of services provided. All installations exist to provide a service, to provide processing efficiently and expeditiously and they should be measured by the extent to which such services are provided in relation to the cost of providing them. Such an approach is meaningful to both user and management. The scope of this paper is limited to aspects of the service provided by a computer centre that relies on the performance of hardware and software, not on availability of facilities, etc. The performance is dependent on actual hardware speeds, operating system characteristics and the reliability thereof in relation to the demand workload.

Service standards in the above context can be expressed as the services that can be expected, on the average and within limits, subject to the demand on resources and demand on those services. The ideal way of setting up the service standards is to balance the standards to suit the actual demand. The selection of these standards can also affect the utilisation of the resources—the well known conflict of optimum utilisation versus ideal service—and as a result standards might be established that compromise between service and utilisation or that attempt to change the workload profile subject to policy decisions regarding more effective use of the resources or from a human engineering point of view. This is where most performance evaluations miss the point. When workload approaches capacity, service levels tend to deteriorate. At this stage it can be seen that not only are optimum utilisation values limited in meaning but also result in incorrect objectives if not qualified as subject to demand. Service criteria are the important factors in planning and operating computer installations and these must of necessity be defined in terms of performance measures. An analytical model is developed in this paper for performance measurement which is directed to supplying that information which relates performance to service.

The cost effectiveness of an installation is usually measured in terms of cost per job or cost per processor hour used or even cost per scheduled hour. A reduction in the first two rates is considered to be an improvement. In this way a minimum unit cost can be determined as well as the unit cost at the desired service level. The difference is referred to as the unit service cost. Management is thus in a position to assess the service standards.

Turnaround model

While many techniques for performance measurement and resource utilisation, ranging through simulation, benchmarks, kernels and instruction mixes, have been proposed in the literature and referred to in a survey (Wiederhold, 1975), together with performance models which emphasise considerations as stability, presented by Mertes (1974), there does not, to the author's knowledge, appear to be much published work which relates performance directly to service standards.

In order to rectify this deficiency, an analytical model, based on service considerations as the main criteria, has been developed. To apply this model in practical situations, information routinely provided by an operating system and controlled experiments are used in the performance evaluation, thereby ensuring that no additional costs are incurred for continual monitoring.

The model described here is postulated on the basis that the most significant measure of service is the turnaround time, i.e. the time taken to complete a job from submission to availability of output. This measure applies to any type of processing, whether submitted by a terminal user or the traditional batch user. An appropriate definition of service standard then is that, for a given category of job, a certain mean turnaround time, with specific variance and limits, is to be provided subject to a specific demand.

In the most general case the major factors affecting turnaround are dependent on the time taken (*a*) to perform the work required for that job, i.e. elapsed time, (*b*) waiting in either the input and output queues, (*c*) to spool input and output, and (*d*) before job is spooled to input queue and after output has been spooled, i.e. manual preparation and clearance of job. The turnaround time for a job will therefore clearly depend on resources available to process that category of job and the number and priority of streams allocated to processing, reading and printing of that class of job hardware and software malfunction, together of course with the personal resources allocated to the manual stages of processing.

Where it can be established that manual processing has either minimal effect on turnaround or is alternatively excluded from the measurement of the turnaround time, the remaining factors can be measured in the normal course of production provided the following information is available for each job: reader, job and writer start times, job end time and the number of records handled or time taken by the reader and writer. The part of total turnaround time that excludes manual processing will therefore be defined as the measurable turnaround time. In other words, it must be accepted that expeditious manual processing is essentially a function of supervision and motivation since it it not possible to describe this often crucial aspect of services in simple mechanical terms.

The measurable turnaround time for a job can be expressed as:

$$t_i = e_i + q_i + o_i + y_i + z_i \qquad (1)$$

where

i = the i'th job
t_i = measurable turnaround time
e_i = elapsed time
q_i = input queue wait time
o_i = output queue wait time
y_i = writer time
z_i = reader time

In this formulation, elapsed time can be regarded as a compound component constituted of further subcomponents. Obviously the time it takes for the work to be done, i.e. processor and input/output (channel and device activity), is an essential part of the elapsed time. In addition, elapsed time of a job contains a wait time subcomponent, i.e. the time that the job is not being processed, owing to system overheads, the fact that other jobs are being processed and any user caused wait times. In other words, the elapsed time can be expressed as:

$$e_i = p_i + f_i + h_i + g_i + d_i + u_i \qquad (2)$$

where

p_i = processor time
f_i = input/output time
h_i = system overhead processor time
g_i = system overhead input/output time

d_i = deactivation time (i.e. when other jobs are running)
u_i = user enforced delay times

The possibility of an overlap of activities generated within any one job, such as an input/output activity concurrent with a processor activity, is not included in Equation (2), since in the author's experience it generally tends to be minimal and requires special monitoring procedures for its measurement. The overlap of activities referred to here relates to that occurring within a job, not that which can take place when the system is run in a multistream mode. The latter form of overlap will be discussed later. In the present context, however, although not included in Equation (2), it would be desirable to measure the extent of overlap in a single stream environment with a hardware monitor and, if significant, insert an appropriate subtractive term, say l_i, into Equation (2).

The formulation in Equation (1) caters for multiple input/output devices and multiple processors, i.e. each of p_i, f_i, h_i and g_i could comprise the time taken on more than one unit. Also, each of the utilisation variables are expressed in such a way that when only one job is being processed, i.e. a single stream environment, a time period can be associated with each of the variables.

The further development of the model makes use of the common practice of categorising jobs with similar resource requirements, qualifying turnaround times accordingly. The mean turnaround time for the k'th job category is then expressed as:

$$T_k = \frac{1}{j_k} \sum_{i=1}^{j_k} t_i \tag{3}$$

where j_k is the number of jobs in that category. The mean turnaround time for all jobs is then:

$$T = \frac{1}{J} \sum_{k=1}^{c} T_k j_k = \frac{1}{J} \sum_{k=1}^{c} \sum_{i=1}^{j_k} t_i = \frac{1}{J} \sum_{i=1}^{J} t_i \tag{4}$$

$$= \frac{1}{J} \sum_{i=1}^{J} (e_i + q_i + o_i + y_i + z_i)$$

$$= \frac{1}{J} (E + Q + O + Y + Z) \tag{5}$$

and

$$\sum_{i=1}^{J} e_i = \sum_{i=1}^{J} (p_i + f_i + h_i + g_i + d_i + u_i)$$

i.e.

$$E = P + F + H + G + U + D \tag{6}$$

where

$$E = \sum_{n=1}^{J} e_i, P = \sum_{n=1}^{J} p_i$$

and so on.

The mean turnaround, both for each job category and for all jobs, can be readily calculated, provided the operation system publishes sufficient information to determine the value of the variables in Equation (5). However, only E, P and F in Equation (6) can be determined from information normally available. The other components of Equation (6) thus require special treatment. These components are therefore discussed in the sections which follow.

User enforced delay times

User enforced delay times (U) can in a batch environment be controlled by minimising delays for activities such as disc and tape mounts and, in fact, for job categories where no operator intervention is required, $U = 0$. For interactive processing it is necessary to determine user delay times as no control exists during production. Some operating systems provide the necessary data. However, where this information is not available, it is necessary to relate results from controlled experiments with the production environment before a reasonable estimate can be made.

System overhead

The real overhead of an operating system cannot be precisely determined as, in the first place, it is not possible to formulate a precise definition of what constitutes overhead and in the second place, in the absence of such a definition, it is not practical to implement a method of measurement. As a result of this lack of precision, overhead is generally simply defined as any work done that is not automatically attributable, from information published by the operating system, to a job run in a single stream environment. This deficiency should be borne in mind when overheads of different operating systems are compared. In these terms, however, some of the elements that constitute overhead are:

(*a*) time taken between activity termination and activity start in a single stream environment, which will be referred to here as initiation/termination overhead, and which refers to both input/output and processor activities

(*b*) the difference in processor time when that processor time which can be directly related to the job under processing is compared with the total processor time used; this time thus represents all overhead functions which require the processor for their execution

(*c*) the increase in processor time, some of which might be allocated to the job itself, when run in multistream environments as compared with that allocated to the same jobs when run in a single stream environment, i.e. those allocations which possibly result from timer inaccuracies, memory and multiactivity management

(*d*) the increased times for input/output activity which results from greater management activity and longer seek times as load increases

(*e*) time taken during step activity in a single stream environment that cannot be directly attributed to the job for either processor or input/output activity.

Response time

For jobs processed in a time sharing environment the turnaround equation reduces to $t_i = e_i$.

$$\text{i.e.} \quad t_i = p_i + f_i + h_i + g_i + d_i + u_i \tag{7}$$

and

$$r_i = \frac{p_i + f_i + h_i + g_i + d_i}{a_i} \tag{8}$$

where r_i is the mean response time and a_i is the number of activities requiring a response from the system. Response time is in fact also applicable to batch processing although it is not generally referred to directly in such terms.

Overlap of activities

An important relationship which strongly influences the quantity of work that can be conducted concurrently in a multiprogrammed environment is the overlap L. Thus the length of a session with scheduled time S can be expressed as a sum of components, as follows:

$$S = P + F + H + G + W - L \tag{9}$$

where

P = total processor time allocated to the jobs
F = total input/output time allocated to the jobs
H = total system processor overhead
G = total system input/output overhead

W = the system idle or wait, i.e. when there is not input/output or processor activity; in a controlled situation W is measurable and in production it is often possible to determine this value from the difference between scheduled time and the system meter readings

L = overlap time.

A suitable procedure for estimating the overlap time can be derived from the work of Snel (1973) and others. These authors have postulated and observed that events for service of I/O devices and CPU tend to be statistically independent. Hence:

$$\rho_{12} = \rho_1 + \rho_2 - \rho_1\rho_2 \tag{10}$$

where ρ_1 and ρ_2 are the probabilities of a processor or channel event occurring and ρ_{12} the probability of either event occurring. If the channel activity can be shown to be independent of the processor activity then it is also reasonable to consider the activity device to be independent of the processor as well. This assumption is essential when use of statistical independence is required to determine the value of some of the variables. The difference beween the calculated and measured values as observed by Snel depend on the number of programs running concurrently. The difference is negative when the system is idle over long periods of time and positive when the system is overloaded. In both the latter cases the events are, however, no longer statistically independent.

Where measurement is not possible, it is essential to cater for those situations in which the events are clearly dependent upon the load, especially where device times are being considered and not channel time, as the times involved are greater. Firstly, the proportion of time a system is idle should be excluded from the total time period. Secondly, when only one event is possible at any one time, such as in a single stream environment without multitasking, the probability of two events occurring concurrently is zero. It can therefore be concluded that the time period during which events can be considered to be statistically independent is the period when at least two independent tasks are competing for those two resources.

This statistical dependence of events also occurs in the overload situation. This arises when a queue for services on a particular device is equal to or greater than the number of tasks currently active. At this stage the processor is waiting for the device and as a result the premise of statistical independence is no longer valid. The increased non-overlapped device time is then based on the time taken per activity and the difference between the average queue length on a device and the multiprogramming factor. This situation can occur on devices where spool, paging or swapping data sets reside.

Multiprogramming level

At this stage there is no indication of the load being placed on the system in conjunction with the values thus far obtained. One of the most significant load measures, given the type of workload, is the number of activities that are conducted concurrently and thus a multiprogramming factor can be defined as the mean number of jobs requiring service from the computer system over a period of time. However, when an interruption is required by a user or an operator, such as for mounting a tape or typing data on a terminal or even the user's think time on a terminal, these periods should be subtracted from the relevant job elapsed times before a multiprogramming factor is determined. On the other hand, the reader and writer (spooling operation) real times should, most probably, be included in the multiprogramming factor calculation. Moreover, for the sake of both compatibility and consistency of definition, the possible gap between each activity terminate and start should be catered for. With the inclusion of the preceding considerations, the multiprogramming factor can be written as:

$$M = \frac{\acute{E} - U}{S} \tag{11}$$

where $\acute{E}$ formally includes the elapsed times of spooling operation and has been corrected to cater for the possible gap between activity termination and start, the gap being based on observations in a single stream environment. The effectiveness of the system can be assessed from a comparison of the multiprogramming factor with two other measures: the overlap ratio and the number of independent units. The overlap ratio is defined as $Z = \frac{L}{S}$. Clearly, $Z < M$ under all situations. The closer Z is to M, the better is the overlapping capability of the system. Also, if the multiprogramming factor is less than the number of units such as processors, devices and channels that are considered significant and independent of each other, it is unlikely that the system is being effectively used, i.e. either it is badly configured in relation to the workload or production policies are incorrect.

Workload

The workload of a system can be universally defined in terms of time taken by the processor and the peripherals no matter what the application. The amount of work processed is dependent on demand, service criteria and resources available. The amount of work processed in a given period is considered to be the supply workload which is not necessarily the same as the demand workload, i.e. the potential workload. This distinction is extremely important when attempts are made at planning and forecasting. If the service standards are based on the demand workload then it is relatively simple to calculate the difference between supply and demand at any stage.

To measure the workload accurately in terms of processor and peripheral time it would be necessary to run each job in a single stream with adequate resources. This is obviously impracticable for a total workload but can be done on a sample basis which can prove vital in performance analysis, as explained later in this paper.

Analysis of performance data

The first step in analysing performance data is to determine the distribution of the multiprogramming factor over the relevant period. This reveals the consistency of operation and focuses attention on those periods of time when low and high multiprogramming factors occur. Thereafter, relationships between load on the system and the variables such as overlap, elapsed time, deactivation, overheads and utilisation can be established once sufficient information has been gathered for various loads on a configuration.

By plotting the mean elapsed time per active stream against the multipgramming factor it is immediately clear at what load level the minimum mean elapsed times occur. Similarly, the behaviour of each variable can be determined for any configuration. Examination of plots of these measurements assists in determining whether hardware and software limitations exist or not.

For example, a graph of the mean processor overhead time per job divided by the mean active stream level plotted against the multiprogramming factor provides a clear picture of the load level at which difficulties can be expected. Some overheads could result from memory management of multiprogramming management overheads. However, with further experiments, it is soon possible to determine the cause of the problem and establish a solution. An alternative representation is obtained by dividing processor overhead by both the scheduled time and multiprogramming factor.

Deactivation and overlap distributions can be similarly

depicted. The increase in deactivation time is dependent on the increase in overhead times and the ability of the system to overlap activities. It is therefore possible to establish what capacity a configuration will attain as a result of a lack of overlapping capability or because system overheads are too great. In the first case minor hardware changes will generally solve the problem. In the second case a thorough investigation of the software is called for since, if the problem does not occur in this area, the only solution is the upgrading of the hardware.

The representations of the state of the system described in the previous paragraphs are of course only valid if the set of jobs J is a valid sample of the population under investigation.

Maintaining desired service standards

Service standards for each job category are dependent on demand for that service and centre policies, subject of course to the capabilities of the system. It is the responsibility of the production supervisors to ensure that these standards are maintained within acceptable limits. A range of limits based on probabilities can be established. This can readily be done provided such standards are defined in terms of measures routinely produced during production.

The variables of the turnaround model presented in the previous sections together with utilisation values form the basis of the reports illustrated in **Tables 1** and **2**. A method of defining service standards is depicted in Table 2. A certain turnaround can be expected, on the average, provided the demand for such a service and the resources used are consistent with those detailed in the table. Most of the information required to determine problems at the production supervisors level is detailed in Table 1. The figures in these tables are hypothetical and are for purposes of illustration only. In Table 1 only the mean values are entered.

If some turnaround standards are not being met, a variation of the demand for such services and the resources used may explain why such a situation has arisen. Alternatively, the cause may be production problems such as excessive input or output queue wait times or excessive elapsed times. Provided however that the mean turnaround of all jobs is comparable with the standard, it can be concluded that a solution to the problem can be achieved by modification of priorities, scheduling principles and job category/stream relationships. Where the mean turnaround of all jobs is not comparable with the standard the explanation may be found in any of the following areas: operation inefficiency, system malfunction and incorrect service standards, as well as in those causes already outlined.

In order to solve problems relating to elapsed time, it is necessary to investigate activity profiles for stream activity, memory and peripheral device utilisation. A number of conclusions can be reached from the stream activity profile (**Table 3**), bearing in mind the desired system activity level and expected multiprogramming factor.

The effect of hardware or software malfunctions can be quantitatively evaluated by determining the amount of stream time not used as compared with the desired multiprogramming factor. A lack of demand, i.e. some or all input queues empty, is an obvious factor to consider. An overloading of resources is clearly depicted in profiles for memory and device utilisation. Where no explanation is forthcoming at this stage, the possibility exists that the cause rests with operation inefficiencies or system design factors such as excessive reserves of critical disc drives or data sets. A report similar in format to that shown in Table 3 for printer activity profile can assist in solving output queue problems.

Improving present service levels

Although the desired service standards are attained, it should be the objective of management to find ways in which better

Table 1 Demand and supply breakdown by job category

Job category (queue)	Number of jobs from prior period	Total no. of jobs	Mean length of queue	%time queue empty	No. of jobs serviced	Mean and Standard Deviation For: Turn-around time (mins)	Elapsed time (mins)	Processor time (secs)	I/O count	Writer time (mins)	Writer count	Reader time (mins)	Reader count	Input queue wait time	Output queue wait time	Memory Size
Student batch (0-5)sec																
Non setup (0-10)sec																
Non setup (10-30)sec	0	200	15	5	200	70	4	20	1500	2	1000	1	400	60	3	90
Non setup (30-120)sec																
Non setup 120sec	4	30	6	0	34	149	20	120	20000	4	5000	2	800	120	3	90
Setup 0-30 sec																
Setup 30-120 secs	0	10	1	30	10	24	10	25	40000	4	5000	2	800	5	3	110
Setup 120 secs																
TOTALS																

Table 2 Service standards

Job category	*Total number of jobs*	*Mean and standard deviation for* *Turnaround time (min)*	*Processor time (sec)*	*I/O count*	*Writer count*	*Reader count*	*Memory size*
Student batch		3	1		100	75	180
(0-5) secs	1,200	3	1	trivial	50	25	0
Non setup		10	2	300	200	150	90
(0-10) secs	480	5	1	200	50	50	30
Non setup		60	12	1,000	1,000	400	90
(10-30) secs	200	30	6	500	1,000	200	30
Non setup		120	25	10,000	3,000	600	90
(30-120) secs	50	60	15	1,000	2,000	300	30
Non setup		300	300	20,000	5,000	800	90
> 120 secs	20	200	150	5,000	2,000	300	30
Setup		90	4	5,000	3,000	500	110
0-30 secs	90	90	4	1,000	2,000	200	50
Setup		240	25	40,000	5,000	300	110
30-120 secs	15	240	15	2,000	2,000	400	50
Setup		300	360	10,000	10,000	2,000	110
> 120 secs	5	300	100	5,000	5,000	600	50
Totals	2,200	24	6, 7	1,200	300	140	100

NB: The times in the 'Job category' column are user estimates, and in columns 'Turnaround time' *et seq*, the first of each pair of entries is the mean and the second the standard deviation of mean

Table 3 Steam activity profile

Stream level	*Percentage time active for* 1 *Stream*	2 *Stream*	3 *Stream*	4 *Stream*	5 *Stream*	3 *Stream***
0	3	0	0	0	0	0
1	96	9	1	8	19	0
2		90	14	1	16	11
3			84	16	4	87
4				72	20	
5					33	

**—See Table 4.

service standards can be established without increasing the cost of production. The main areas in which improvements can be effected are software system tuning and in the service standards themselves, as well as in those areas previously mentioned, i.e. operations efficiency, optimum stream level, job category/stream relationship, priorities and scheduling algorithms.

Optimum service standards, subject to demand, organisation policies and resources available, can be determined by using optimisation techniques on the turnaround model. It is postulated that with the use of linear programming techniques it is possible to optimise the following expression:

$$\sum_{k=1}^{c} t_k X_k$$

where

$$X_k = \frac{j_k}{J},$$

$$0 < X_k < 1$$

and

$$\sum_{k=1}^{c} X_k = 1$$

X_k represents the proportion of jobs in the k'th job category as compared with the total number of jobs and t_k is the mean turnaround time for the k'th job category.

Optimum multiprogramming

Controlled experiments have been conducted to determine the multiprogramming factor with the aid of the turnaround model. The latter cannot be determined in the production environment since the optimum factor is a function of workload variations, available resources, e.g. tape drives, memory, etc. and the ability of the system to multiprogram. Controlled experiments, in the form of a benchmark representing a sample of the demand workload with 134 jobs and 315 activities (steps), were conducted with the number of permissible streams varying from one to five on an IBM 370/145 with 384K bytes using OS/VS1 Operating system (**Table 4**).

The resource utilisation figures obtained for the single stream experiment were assumed to be applicable to the multistreamed experiments as well (job processor time and I/O activity in Table 4). Any variation of those figures were regarded as a portion of the increased overheads. It was also assumed that the overheads for the single stream experiment are at a minimum with little or no memory management (this must be checked in a paging environment) or multiprogramming management required (adjusted figures for a single stream in Table 4). The benchmark jobstream had previously been checked for the extent of processor and channel overlap in a single stream environment and these factors were found to exert negligible effect on the outcome of the experiment when

Table 4 Summary of multistream experiments

Stream level	*Mean stream activity*	*Multi programming factor M*	*Step elapsed time*	*Elapsed time E*	*Scheduled time S*	*Writer time*	*Overhead processor time H*	*Overhead I-O Activity time G*	*Overlapped time L*	*Deactivation time D*
1	1	1,32	13,364	14,076	14,076	4,552	2,631	3,534	1,200	—
1*	1	1,42	—	10,899	10,899	4,552				
2	1,95	2,35	23,284	23,819	12,243	4,928	3,031	5,569	5,468	6,108
3	2,85	3,29	34,239	34,774	12,191	5,384	3,476	6,374	6,658	15,813
4	3,56	3,99	43,708	44,243	12,419	5,328	3,627	5,677	5,995	25,828
5	3,46	3,90	42,166	42,701	12,325	5,407	3,910	4,948	5,593	29,704
3**	2,87	3,28	34,024	34,559	12,036	4,883	3,722	5,639	6,435	16,088

Job processor time $P = 4138{\cdot}54$ seconds and Job input/output time $F = 4972$ seconds
*Single stream figures adjusted to reflect no paging assuming 25 ms per EXCP
**Input/output activity spread over two devices.
All times are published in seconds.

Table 5 Memory upgrade experiments
N.B. Same correction was applied as in the previous study

		Single stream		*Two streams*		*Three streams*
		256K	384K	256K	384K	384K
Pages (No of)		176,850	110,506	458,605	199,307	225,817
Step elapsed time	(sec)	16,218	13,317	34,359	22,762	31,486
Job processor time	,,	4,072	3,688	3,941	3,766	3,892
Total processor time	,,	7,012	6,630	8,970	7,332	7,784
Scheduled time	,,	18,872	15,693	19,527	12,513	13,956
Corrected elapsed time	,,	18,872	15,693	37,013	25,138	33,862
Writer time	,,	3,077	2,680	3,035	2,451	2,916
Multiprogramming factor		1,16	1,17	2,05	2,20	2,64

no paging occurred.

In the preparation of the figures, a number of adjustments were made. Thus, it was assumed that, provided there is no system idle time during the experiment, elapsed time of the jobs must be equal to the length of the session and the multiprogramming factor correspondingly is equal to one. This adjustment is necessary since the system publishes start and terminate times at a convenient location in its logic. For multistreamed experiments, the elapsed time gap for the single stream is merely added to the elapsed times. This is a valid approach provided such activities have a high priority and they are not directly affected by memory management.

Conversion of input/output activities to input/output time used on various peripheral devices, together with the proportion of processor time used during initiation and termination phases in a single stream environment, is based on factors obtained from special test runs in that environment. These values are required to solve the turnaround model equations for the single stream benchmark run. The utilisation values (processor time P and input/output time F) thus obtained are used in the solution of equations for all multistreamed experiments.

A minimum value of the input/output overhead is known for paging and initiation and termination activities. However, overlap can be estimated and, in conjunction with the scheduled time equation (9), a feasible total input/output overhead time is obtained. The extent to which overlap occurs is estimated by assuming statistical independence of events on the heavily used direct access storage device (these experiments were conducted using only one device for most of the system activity) and the processor over the period when more than one activity is taking place, i.e.

$$L = \frac{P + H}{S} \cdot \frac{Fd + Cd}{S} S'$$

where $P + H$ is the total processor time and $Fd + Cd$ is the total device input/output time on the particular critical device and S' is the adjusted scheduled time.

A graph (**Fig. 1**) of the mean elapsed time per active stream of all the jobs as a function of the multiprogramming factor based on the above controlled experiments resulted in a horizontal straight line for more streams than 1 but should result in a concave curve, provided the range of multiprogramming achieved is sufficient to include an overload situation. At the time of writing, only the results of a three stream experiment, with paging data sets spread over two devices, was available. The mean elapsed time per active stream is the same as for paging on a single device. In this case the reason is obvious as the amount of memory and the paging algorithm do not lend themselves to overloading other resources. The critical resource is consequently memory and the optimum multiprogramming factor is when two streams are active as this allows for maximum flexibility but, at the same time, minimum job elapsed times.

Predicting the effects of change

The effect of change of either the workload or resources must be quantified to determine cost effective upgrades. Performance improvements of a memory upgrade from 256K to 384K on an IBM 370/145 was predicted by making use of controlled experiments and the analytical model. Results from controlled experiments after the memory upgrade compared favourably with the predictions.

The maximum demand for memory can be determined by summing the memory requirement for each step, weighted by the elapsed time for that step, in a single stream environment. The manner of access of that memory and the proportion that is relatively frequently required is not known. From the benchmark runs it was possible to determine the overhead of both processor and I/O resources as a function of workload memory required and available (**Figs. 2** and **3**). The values on the graphs were derived from single and two stream

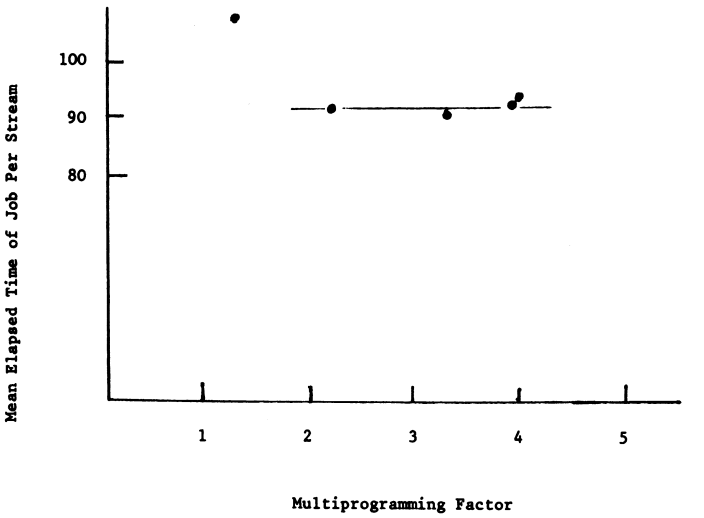

Fig. 1

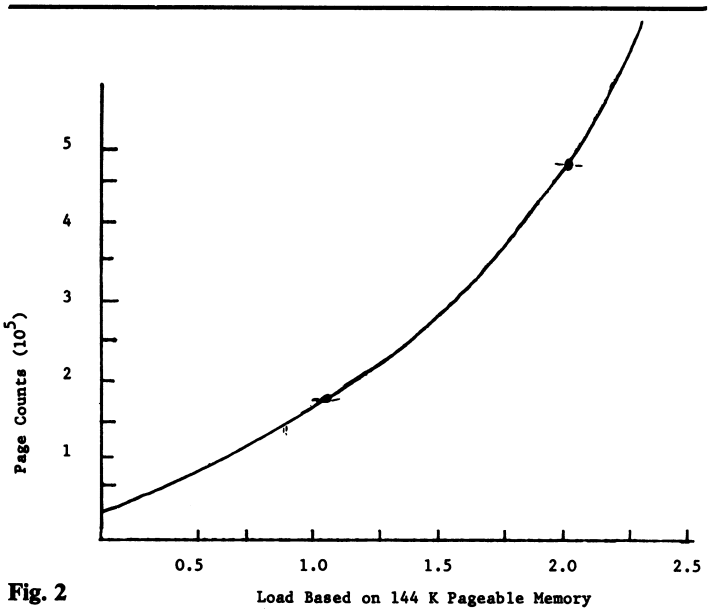

Fig. 2

experiments on the IBM 370/145 with 256K (**Table 5**).

The real pageable memory of 144K was used as a base and a single stream experiment was assumed to have a factor of 1·16, i.e. a multiprogramming factor, whereas the two stream experiment resulted in a factor of 2·05. The page count was expected to be greater than zero, due to memory access method, and the processor time must tend to a value which caters for initiation and termination overheads, normal overheads while processing and the actual work time. Based on experiments with no paging requirements, the system overhead was approximately 66% of the processor work done.

With the introduction of another 128K of memory the available real paging area of 272K results in a decreased load factor of $\frac{144}{272} = 0{\cdot}53$. With the use of this factor the paging count and processor time for any loading on the new configuration can be determined. Experiments with one, two and three streams active were run for which the forecast page counts were 98,000, 230,000 and 220,000 while the total processor times were 6,425, 7,350 and 7,700 seconds respectively.

With the use of the turnaround model it is possible to predict the scheduled time based on a constant multiprogramming factor and consequently the improved throughput ratio (work done) as compared with the reduced scheduled time. Consider, for example, the two stream environment with a total processor time of 7,000 and a page count of 160,000. The total input/output time for work done in a single stream was 1,877 seconds on tape drives, 965 seconds on 2,314 direct access devices and 2,130 seconds on 3,330 direct access devices, i.e. $F = 4{,}972$ seconds. From the table, the difference between scheduled time and step elapsed time in a single stream run is 2,654 seconds, i.e. step and job initiation and termination overheads. Assuming a 20% processor time activity during this period, the input/output overhead is given by

$$G = 2{,}654 - 531 + (160{,}000 \times 0{\cdot}025) = 2{,}123 + 4{,}000 = 6{,}123 \text{ seconds}$$

The overlap probability for the same load, i.e. multiprogramming factor, should be consistent, hence for the two stream run the overlap probability is

$$\frac{L}{S} = \frac{1}{S}(P + F + H + G - 1)$$

$$= \frac{1}{19{,}527}(8{,}970 + 4{,}972 + 2{,}123 + 11{,}465) - 1 = 1{\cdot}41 - 1{\cdot}0 = 0{\cdot}41$$

The new scheduled time can now be estimated, viz.

$$S = P + F + H + G - 0{\cdot}41$$

i.e. $$1{\cdot}41 = 7{,}000 + 4{,}972 + 6{,}123 = 18{,}095$$

i.e. $$S = 12{,}833 \text{ seconds}$$

Based on this estimate the throughput improvement, in terms of job processor utilisation in a single stream, with the same load, is 52%.

To confirm the validity of statistical independence of activities on a device and the processor, the overlap value for each experiment is compared when derived from the scheduled time equation and calculated statistically. The device time on 3,330 system resident packs is 1,726 seconds for work done, as accounted to the jobs. This time together with the paging activities on the same disc packs forms the major portion of the device time that can be overlapped with the processor time although each device (tape, 2,314) is accounted for separately. Where the multiprogramming factor is approximately 2 or more, then the initiation and termination input/output overhead (2,123 seconds) must be considered as well. A less than 5% difference in the overlap times was found to exist for the two and three stream experiments.

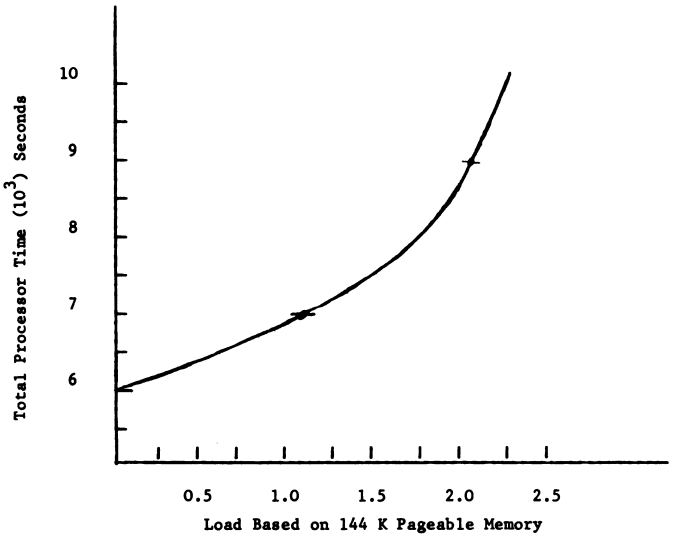

Fig. 3

Conclusion

The turnaround model described here embraces aspects from utilisation to service standards and consequently provides a

coherent analytical tool for various types of analysis. It can be used to control and measure day-to-day operations environment on the one hand and forecast the effect of alterations to configurations on the other. It also provides a reasonable initial basis for system evaluation and isolation of those areas which may require further investigation with the aid of appropriate monitors.

The model has already been used to assist management of a centre in the selection of computer equipment for both batch processing and time sharing services. It has also been used at existing installations to determine stream levels that result in a minimum mean elapsed time of jobs and predict the effect of configuration changes such as direct access storage and memory upgrades. In these applications, techniques derived from the model have been of useful practical assistance.

The turnaround model proposed here requires, of course, considerable refinement. Obviously, also, introduction of optimisation techniques would considerably enhance the value of results derived from the model. The author hopes, however, that this description of the model and its application at some existing installations will be of assistance to others faced with the all too familiar problems of computer centre managers.

References

KIMBLETON, S. R. (1973). Performance Evaluation—Directions and Implications, Remarks Delivered at the First SIGME Symposium, *Performance Evaluation Review*, Vol. 2 No. 1.

MERTES, L. H. (1974). The Top Data Processing Manager's View of the Role of Operations Management, *Share XLIII*, pp. 692.

SCHUMACHER, D. (1973). A Methodology to Establish and Control Computer Performance, *Share XLI*, pp. 751.

SNEL, D. (1973). Computer Performance Measurement and Evaluation Tools, *Share Measurement and Evaluation*, Vol. 2, pp. 210-219.

TIMMRECK, E. M. (1973). Computer Selection Methodology, *Computing Surveys*, Vol. 5 No. 4.

WIEDERHOLD, J. F. A. (1975). Selection of computer measurement and evaluation techniques, *Systems*, September 1975.

Book reviews

Computer Data Structures, by J. L. Pfaltz, 1977; 446 pages. (*McGraw-Hill*, £14·60)

Data Structures and Programming Techniques, by H. H. Maurer, 1977; 228 pages. (*Prentice-Hall*, £10·85)

It can be argued that detailed consideration of data structure is an 'academic' subject, which has little to do with the reality of day-to-day commercial data processing. Pursuit of such an argument tends to rob programming practitioners of some useful tools. Anyone involved in the specification or design of processes to handle large quantities of related data should spare some effort to study suitable texts and to work through suitable examples. Programmers will recognise that the efficiency, clarity and complexity of any program purporting to solve any given problem depends to a large extent upon the selected data structure. Indeed there is a popular view that the ability to describe formally the data structures involved is a necessary prerequisite to the solution of any such problem.

Both of these books begin by introducing their own formal notation, based upon graph theory and set theory respectively. All necessary manipulations are described and demonstrated in a simple way to enable the reader to practise and understand the notation used. Beyond this point the two books differ in both style and emphasis.

Maurer's text, which has been extremely effectively translated from the German by Camille Price, has a more theoretical approach. Most of the structures described are assumed to exist within the knowledge of the reader. The book formally defines each structure and illustrates its properties by means of a mathematical analysis. An example PL/I program is given to illustrate a process to manipulate the structure; this technique represents solutions in search of problems!

Computer Data Structures has a better approach. After the essential notational introduction, abstract structures are introduced as necessary to solve particular problems. Several possible computer representations of each abstract structure are given, with a detailed description of the merits of each. Using formal graph theory, techniques for describing the nature and efficiency of data structures are developed. The problem areas covered include interactive graphics, dynamic storage allocation, virtual memories and file structures. Throughout the text procedures are defined in an obvious ALGOL-like language, but can easily be realised in FORTRAN, PL/I or other languages; necessary guidance is given to enable those processes which are essentially recursive to be implemented non-recursively.

Clearly, I personally favour the book by Pfaltz, as being more suitable for both students and practitioners of computer science. In general its diagrams are better and more meaningful, the worked examples are more helpful and the 'excercises for the reader' more likely to stimulate one into actually trying them out.

ALAN CHANTLER (Yelvertoft)

Computing in Clinical Laboratories, edited by F. Siemazko, 1978; 302 pages. (*Pitman Medical*, £10)

The proceedings of the second international conference on computing in clinical laboratories was held in Birmingham in September 1977 and the papers presented are now available in hardback form. The book is subdivided into sections on system design and implementation, microprocessors, cost effectiveness, experience of input/output devices, interfacing and remote processing, computer assisted choice and use of laboratory results and, finally, recent advances in numeric techniques.

Nowadays, no book can cover completely as extensive a subject as clinical laboratory computing. However, most of the basic issues are explored in the 33 papers from centres across Europe and beyond and most papers list references to enable the topic to be explored in more detail. I particularly appreciated the growing interest in the analysis and use of laboratory data. The utilisation of the standard techniques of data analysis to explore the meaning of the enormous volume of material produced by automated laboratory systems is crucial to improvements in system design and use; as well as trend analysis and discriminate analysis, the appearance of payoff matrices, utility functions and especially decision trees augurs well for future developments in the exploration of investigation strategies and medical decision making.

In addition to the data analysis, about a quarter of the book is devoted to the exploration of the implications of microprocessor technology for laboratory computing. This is a rapidly developing area. Microprocessors are likely to be included in most items of analytical equipment and a balance needs to be struck between the integral processing capacity of the laboratory equipment and the data processing activity of the total laboratory (or hospital) system. The evaluation of cost effectiveness of laboratory computing continues to be explored but this still presents difficulties in the absence of serious evaluation of laboratory information within the medical system.

The book provides a useful exploration of the present state of laboratory computing.

BARRY BARBER (London)