

Validation of an analytic model of computer performance

E. Foxley† and O. Salman‡

The objective of validating an analytic model for performance prediction of a computer system is to establish confidence in its use as a tool. In general the problem of validation has two main aspects. The first deals with the 'internal correctness' of the model, in that its mathematical structure must satisfy the functions and logical sequences intended by the designer. This should be a basic objective of the analyst, and is the simpler of the two aspects. However, the more formidable aspect of validation lies in deciding how adequately the model represents actual systems and how far it reflects their major characteristics. The work in this paper deals with this latter aspect. A procedure for estimating input parameter values and output performance measures from available performance statistics and measurements is described. The model used is a class of queueing networks known as central-server, although the same procedure could be applied to other types of model.

(Received May 1980)

1. Introduction

Two techniques could be used in order to validate an analytic model (Salman, 1978):

- by cross correlation of the model's results with either a comparable simulation model, or another analytic one of the same system developed by using other mathematical tools;
- by comparing parameters and results of the model with actual measurements and data of available computer systems.

The first method of validation is most useful when the system to be modelled is in the design or planning stage, but in the absence of any measurements there will be no guarantee that the model is valid.

On the other hand, if there is at least one implementation of the modelled system for which historical data and measurements are available, validation of the model could then be achieved by comparing the calculated performance values with the corresponding measurements for the same values of (input) parameters. This is the most accurate method of validation.

However, validation by this technique is not a straightforward one. In contrast to comparison with a simulation model, for example, the data collected from virtually all measurement monitors and performance packages of available systems are not generally in that 'basic' form required by analytic models. Usually further processing on these data is needed before they can be used in the validation process. The data collected from available computers are usually produced by hardware monitors and/or operating systems monitoring packages. The parameter values required to validate the analytic model have to be extracted from these data, from the workload specifications and from any other relevant information about the system. Experiments specifically designed for this sort of validation will always be required (Berners-Lee, 1972; Giammo, 1976; Salman 1978).

2. The model

The model is a queueing network, and its main features are shown in Fig. 1. This type is known as a central-server network. It is a popular one among analysts since it can generally represent the main structure of most typical modern computer

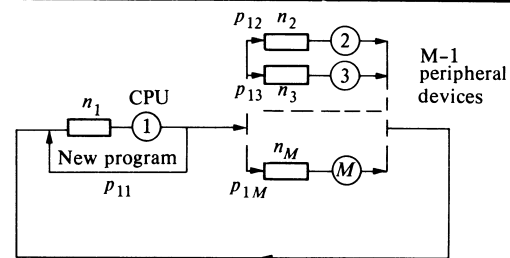


Fig. 1 Schematic description of the central-server queueing network

systems. Server 1 represents the CPU, and servers 2, 3, . . . M represent I/O units. The model has the following assumptions:

- (1) the number of multiprogrammed jobs in the system is constant (constant degree of multiprogramming);
- (2) the routing probabilities are independent of the state of the system;
- (3) the service times are independent of the state of the system;
- (4) there is no concurrent processing, i.e. only one resource is allocated to any one job at a time.

Analyses for such a model have been carried out by several authors (Buzen, 1971; Chandy, 1972; Chandy *et al.*, 1975a; 1975b; Gelenbe, 1975; Gelenbe and Pujolle, 1977; Gordon and Newell, 1967; Kobayashi and Reiser, 1974; Moore, 1971) and formulae for several performance measures have been established. In its final analysis the model has the following input parameters and output measures:

Input parameters

M the number of resources in the system

N the number of jobs being multiprogrammed

μ_i^{-1} the mean service time per job at the i th resource ($i = 1, 2, \dots, M$)

σ_i^2 the variance of service time per job at the i th resource ($i = 1, 2, \dots, M$)

Output measures

The mean and variance of queue lengths at the different resources ($\bar{n}_i, \sigma_{n_i}^2; i = 1, 2, \dots, M$)

The mean delay at each resource

†Department of Mathematics, University of Nottingham, University Park, Nottingham NG7 2RD, UK.

‡Department of Computer Science, ISSR, Cairo University, Egypt.

The utilisation of each resource

The average total time in the system per job

The reader who is particularly interested in the mathematical analysis of the central-server model should refer to Buzen (1973), Kobayashi and Reiser (1974) or Salman (1978).

3. The validation procedure

The machine used for the validation procedure was an ICL 1906A running under the GEORGE 3 operating system.

When GEORGE 3 is running, various relevant performance statistics are gathered by a special package (which is an integral part of the operating system), and stored in a system file. The frequency with which the data are collected is controlled by an installation parameter. The data which have been collected can then be read by other programs to analyse them in any required way. Performance statistics were collected for a whole week. Intervals which cover all the working shifts (work was done in three shifts) over 24 h were selected. These were

Table 1 Nottingham University GEORGE 3 performance statistics for 8 September 1977 on 1906A

(a) At 21.37.58

Time	Jobs			Core image details					Core usage (Kwords)				CPU time (% of clock)			
	MOP ^a	BKG ^b	Total no.	>0K ^c	>2K	>35K	>70K	>100K	OBJ ^d	GEO ^e	FREE	TOT	OBJ	GEO	IDLE	EXEC ^f
21.43	2	14	15	2	10	2	1	0	156	29	5	190	85	9	1	5
21.45	2	14	14	2	10	2	0	0	149	34	7	190	67	16	6	11
21.47	2	16	17	2	12	3	0	0	126	52	6	184	36	38	13	13
21.49	1	15	16	2	11	3	0	0	102	55	32	189	38	26	26	10
21.51	1	15	15	1	10	4	0	0	152	30	8	189	36	27	27	10
21.53	1	13	14	1	10	3	0	0	59	25	64	149	46	30	11	13
21.55	1	13	14	1	10	3	0	0	16	109	68	192	25	7	64	4
21.57	1	14	15	1	10	4	0	0	139	40	8	186	56	22	13	9
21.59	1	14	14	2	10	2	0	0	141	38	9	188	14	27	51	8
22.01	1	12	13	1	10	2	0	0	161	24	6	191	44	26	19	11
22.03	1	13	13	1	10	2	0	0	43	90	38	171	43	20	30	7
22.05	1	14	15	1	12	2	0	0	137	41	6	184	37	26	26	11
22.07	1	13	14	2	10	2	0	0	121	58	9	188	57	18	17	8
22.09	1	14	15	1	11	2	0	1	134	46	7	188	61	27	5	7
22.11	1	13	13	1	10	1	0	1	47	96	48	191	24	34	32	10
22.13	2	14	14	1	11	2	0	0	137	23	8	168	46	32	13	9
22.15	1	14	15	2	12	1	0	0	157	22	6	185	58	30	3	9
22.17	2	14	15	1	12	2	0	0	130	51	9	190	68	21	3	8
22.19	2	14	15	1	12	2	0	0	160	15	7	182	72	18	1	9
22.21	2	13	14	1	10	3	0	0	125	25	41	190	68	21	3	8
22.23	2	13	14	1	10	3	0	0	149	32	5	187	51	31	10	8
22.25	2	13	14	1	11	2	0	0	161	25	6	191	52	37	0	11
22.27	2	15	15	1	12	2	0	0	141	39	9	189	76	18	0	6

(b) At 00.00.50—backing store device transfers queued

Time	36 ^g	37 ^g	42 ^h	43 ^h	44 ^h	45 ^h	46 ^h	52 ⁱ	57 ⁱ
00.16	6								
00.18					2	2			1
00.20							1	2	2
00.22			1					1	1
00.24	1								
00.26				1		1			
00.28							2		3
00.30	2		1			1	6	2	
00.32	2								
00.34					2		1		
00.36	1		2			3			
00.38	3						1	3	
00.40									
00.42	2								
00.44						1			
00.46				1	6			2	
00.48									
00.50	1		4						
00.52			1	1		4			

^aMOP—multiple online programming.

^bBKG—background.

^cK = 1024 words, 24 bits each.

^dOBJ—object.

^eGEO—GEORGE.

(c) At 07.14.54—backing store device transfers per minute

Time	36	37	42	43	44	45	46	52	57
07.14	424	374	158	1	100	50	144	644	171
07.16	877	397	248	0	34	216	287	141	141
07.18	1201	467	348	39	50	97	305	5	289
07.20	1102	298	254	0	126	76	33	31	580
07.22	691	839	230	31	193	198	105	412	103
07.24	408	359	145	13	169	24	51	173	31
07.26	987	403	288	14	43	2	43	37	13
07.28	252	78	63	0	12	0	59	15	299
07.30	193	82	64	0	12	16	5	153	38
07.32	791	411	237	3	73	19	133	6	39
07.34	2319	636	562	10	189	116	277	62	171
07.36	1479	985	539	26	281	218	110	117	154
07.38	395	908	226	21	649	455	46	246	331
07.40	822	717	355	9	650	273	109	69	256
07.42	387	819	215	19	179	444	229	59	131
07.44	934	359	223	14	416	190	160	25	223

^fEXEC—executive.

^gDevices 36, 37 — 2 HSD connected to one controller.

^hDevices 42–46 — 5 EDS60 connected to one controller.

ⁱDevices 52, 57 — 2 EDS60 connected to one controller.

periods during which the machine was running steadily and no breakdown occurred. The statistics collected by the performance package are mainly snapshots of the overall state of the system at a given instant; these were taken every 2 min. The information collected at each snapshot comprises items such as: CPU idle time and number of transfers since last snapshot, main store state, queue length at each resource, and the type (foreground or background) and number of jobs in main store (Table 1). With the exception of the CPU idle time and number of transfers all the other information is non-cumulative. It gives knowledge about what is happening at the instant of the snapshot and no indication about anything happening in the time between snapshots.

4. Estimation of input parameters

In what follows we discuss a procedure to estimate the input parameters described in Section 2.1.

4.1 The number of system resources, M

Although this value represents the number of different resources of the system, the value chosen will depend upon the level of detail required by the analyst. For example, the configuration used has one CPU and three controllers connected respectively to the following devices: two exchangeable disc stores (EDS), five EDS and two high speed drums (HSD). This will give M a maximum value of $1 + 2 + 5 + 2 = 10$ (devices) and a minimum value of $1 + 1 + 1 + 1 = 4$ (controllers). It will be easier to consider these resources at the controller level, i.e. to use the minimum value of M , so long as we can estimate the corresponding service times.

4.2 The number of multiprogrammed jobs in the system, N

This is the number of jobs resident in memory (main store) during the interval under study. The number of such jobs is called the degree of multiprogramming which the model assumes to be constant. In general, this is not true unless, in the system used for validation, two conditions are met. First, the memory has a fixed number of partitions with each one allocated to one job at a time. Second, the system is heavily loaded so that there is always a backlog of jobs, each ready to occupy immediately any partition that is freed by a completed job. In practice, there are comparatively few systems which still comply with the first condition, the best known being the IBM operating system MFT. However, by examining Table 1(a) we can see that the number of jobs reported in memory over an interval of 28 min determined by the snapshots at 21:51 to 22:17, varied only between 13 and 15. (The reader should note that since each snapshot includes information accumulated during the preceding 2 min, the above interval of 28 min starts 2 min before the snapshot at 21:51, i.e. at the moment following the snapshot at 21:49. Thus, without further prompting, any reference to an interval specified by a number of snapshots is always assumed to include the 2 min preceding the first snapshot.) A characteristic shown by all the performance statistics over a whole week was the small variation in the number of jobs in memory. There was never any dramatic change in the number of jobs multiprogrammed over a substantial period of time, even though the system memory was not statically partitioned and any program was allocated a partition of storage in proportion to its requirements. In fact this is not the first time this phenomenon has been reported. Anderson *et al.* (1973), in an empirical study of time sharing systems, reported that the degree of multiprogramming remains stable around 4 or 5 over a long period of time when the number of users logged on was over 25 or so. This means that our assumption of constant degree of multiprogramming is reasonable provided that the system is heavily loaded, that is, provided that the second condition at least (and not neces-

sarily the first) is met. To estimate an average value for N over a certain interval of time we have to consider both the number of user jobs in main store and the effect of the operating system. It becomes necessary to distinguish carefully between the concept of degree of multiprogramming as used to signify the number of simultaneously active user jobs, and its use in modelling to represent the total extent of parallel activities on the machine including both user jobs and the operating system. Our estimated value of N in the model should therefore include the demands made by the operating system on the CPU and I/O in the same way as those of the user jobs. Since the demands made by the operating system are more than what is usually generated by an average user job, a factor F ($F > 1$) should be added to the average number of user jobs over the period under study. In effect the value of F represents the average number of parallel processes generated by the operating system. Thus an estimate for N can be given by

$$N \approx \text{round} \left\{ \frac{\sum_{i=1}^s J_i \Delta T_i}{\sum_{i=1}^s \Delta T_i} + F \right\}$$

where $\text{round} \{X\}$ gives the integer value nearest to X , J_i is the number of user jobs in core at the i th snapshot ($i = 1, 2, \dots, S$); ΔT_i is the time interval between snapshots ($i - 1$) and i ($i = 1, 2, 3, \dots, S$); and F is a factor dependent on the degree of parallelism within the operating system.

In the validation procedure described below, F has been assumed to take the value 2. This has given reasonably good results with GEORGE 3; different values would need to be considered for other operating systems.

4.3 The routing probabilities (Matrix P)

In the routing matrix

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & \dots & p_{1M} \\ 1 & 0 & 0 & \dots & 0 \\ 1 & 0 & 0 & \dots & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & 0 & 0 & \dots & 0 \end{bmatrix}$$

only the elements of the first row ($p_{11}, p_{12}, \dots, p_{1M}$) have to be evaluated, as the rest are determined by the model's structure. The values of these elements, in any interval under study, could be estimated from the number of transfers recorded in the performance statistics over this interval. Thus, if f_i is the average number of transfers per job to the i th I/O resource

($i = 2, 3, \dots, M$) then $\sum_{i=2}^M f_i$ is the average number of transfers

per job. Now if we let $f_1 = 1$, then $f = \sum_{i=1}^M f_i$ is the average

number of CPU requests per job, where the choice of value for f_1 accounts for the last CPU request before the job is completed and leaves the system. Hence, an estimation for p_{1i} is given by

$$p_{1i} = \frac{f_i}{f} \quad i = 1, 2, \dots, M$$

Assumption (2) of Section 2 states that these probabilities are constant and independent of the state of the system during the period under study. To see how far this statement is valid, the performance statistics at the interval between 09:50 and 10:12 h (i.e. the 24 min following the snapshot at 9:48) were examined. The values of f_i (in this case $i = 1, 2, 3, 4$) were calculated for different subintervals within the 24 min and the

corresponding p_{1i} obtained. If these p_{1i} were to remain stable over this interval, then by selecting any two (unoverlapped) sub-intervals we would expect both to have the same (or nearly the same) values of these probabilities. For example, by calculating the values of p_{1i} ($i \sim 1, 2, 3, 4$) over the three subintervals of lengths 8, 10 and 6 min delimited by the snapshots at (09:50, 09:56), (09:58, 10:06) and (10:08, 10:12) respectively, it can be confirmed that p_{1i} is reasonably stable (see Table 2). Moreover, it was also found that sometimes these probabilities could be stable over substantial periods of a whole shift. This is illustrated in Table 2 by the values of p_{1i} evaluated at the time interval 11:27–11:35, i.e. ~ 1 h later. We thus come to the conclusion that the assumption of constant routing probabilities P independent of the system state is, in fact, a realistic one.

Table 2 Elements of the first row of matrix P

Time interval	Interval length (min)	P_{1i} of matrix P			
		p_{11}	p_{12}	p_{13}	p_{14}
09.50–09.56	8	0.0041	0.5900	0.2801	0.1258
09.58–10.06	10	0.0043	0.5810	0.3077	0.1052
10.08–10.12	6	0.0040	0.6100	0.2910	0.0950
11.27–11.35	10	0.0045	0.5738	0.2837	0.1380

4.4 The mean service times (vector \mathbf{m})

The mean service times of the different resources in the system are represented by the vector

$$\mathbf{m} = \left(\frac{1}{\mu_1}, \frac{1}{\mu_2}, \dots, \frac{1}{\mu_M} \right)$$

We shall examine how to estimate the mean service of each type of resource.

(a) The processor (CPU)

An estimate of the CPU mean time $1/\mu_1$ could be worked out as follows:

Suppose that the system has been examined for a period of time T min. Without loss of generality, T will be taken as a multiple of the time interval between the snapshots; in our particular example this is 2 min. Let u_1 be the average CPU utilisation over this interval T . The value of u_1 could be estimated from the CPU busy times as recorded at the snapshots of the performance monitor over the interval T [see Table 1(a)]. Hence the CPU actual busy time during T is $u_1 T$. If N is the average number of jobs in time T as explained in Section 4.2 we have

$$\frac{1}{\mu_1} = \frac{u_1 T}{Nf} = p_{11} \frac{u_1 T}{N}$$

(b) The I/O mean service time

This is dependent on the type of resource used. We discuss the two most commonly used devices—EDS and HSD.

Exchangeable disc store. In general let us assume that one controller is connected to p spindles [Fig. 2(b)]. The timing diagram for one spindle is shown in Fig. 2(a). It is possible to determine analytically the distribution of the response time for this file system. Similar systems have been studied mathematically by Abate *et al.* (1968) and also by Seaman *et al.* (1966). However, since an estimation of the average response time (or mean service time) of such a system is all that is required here, a much simpler analysis could be carried out as follows.

Let us assume that all the p spindles are similar, in the sense that they all have the same timing diagram with the same

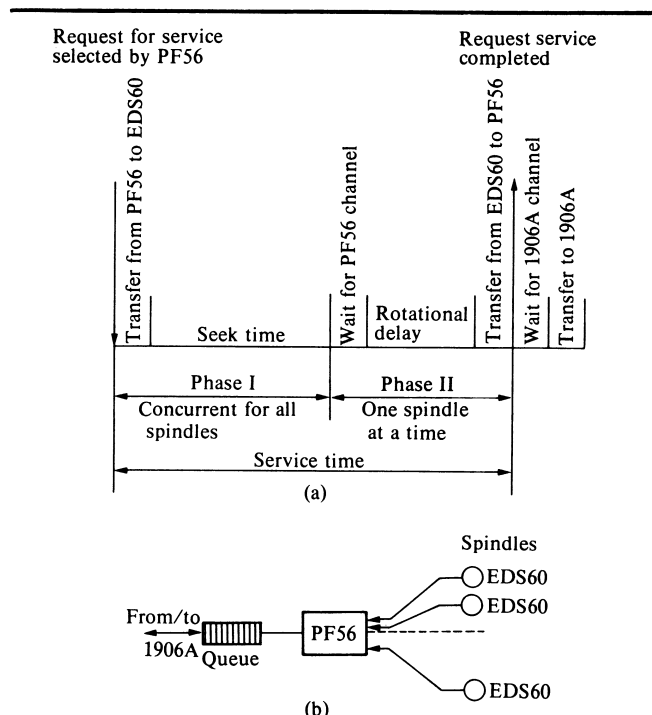


Fig. 2 (a) The phases of execution of a disc (EDS60) service request, (b) multiple spindle controller queueing model (main computer—ICL 1906A; controller—PF56; spindle—EDS60)

average values of these timings. This is true in most physical structures. We also assume that each spindle receives requests at a rate which is the same fraction of input for all spindles, i.e. $1/p$ of all requests for access received by the control unit. This means that the file records are assumed to be distributed uniformly over all the p spindles. Essentially that is not what occurs in practice, as the spindles which happen to store the frequently accessed files tend to get a bigger share of the traffic. However, most computer installations try to redistribute their files so that the traffic becomes more balanced. This situation is generally desirable as it permits fuller utilisation of all access mechanisms and the overlap of seek operations. Now, if t_s and t_d are respectively the average seek time and latency per spindle, and if t_{\max} and t_{\min} are respectively the maximum and minimum access time to this filing system, then

$$t_{\max} = t_d + t_s$$

and

$$t_{\min} = t_d + \frac{t_s}{p}$$

Hence, an estimation of the average access time is

$$t = \frac{t_{\max} + t_{\min}}{2} = t_d + \frac{1}{2} \left(1 + \frac{1}{p} \right) t_s$$

where t_d is usually taken to be one half of the time of one spindle revolution. For example, one of the three controllers in the system used for validation was connected to five EDS60 spindles each with average seek of 35 ms and average latency of 12.5 ms. For this device, the formula gives a $1/\mu$ of 33.5 ms (5.58×10^{-4} min).

High speed drum. The timing diagram for a HSD is shown in Fig. 3. Here there is no arm movement, i.e. no seek time as in discs. Most of the time needed to complete a service is attributable to the rotational delay, i.e. the latency. Whatever the number of drums connected in parallel to the same controller, a request service to one of the drums has to wait for the whole

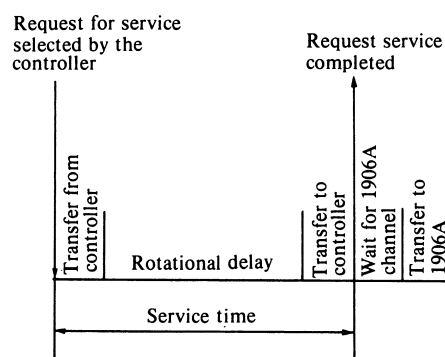


Fig. 3 Timing diagram for a HSD

of its own rotational delay before transmission (usually taking a very small time) starts. This means that the average service time of one set of p drums connected to the same controller, is the same as the average latency of one drum independent of p . Of course, this is on the assumption that these drums are exactly of the same type, which is a usual common practice. Thus the mean service time ($1/\mu$) of a cluster of drums on one controller is given by the average latency of one drum which is half the time of one revolution.

The revolution time for the HSD 2815 used in the ICL 1906A, is 12.6 ms, giving a $1/\mu$ of 6.3 ms (1.05×10^{-4} min).

4.5 The workload

Usually the workload on any machine will tend to vary over time. We will assume that the workload consists of either a

homogeneous stream of jobs (e.g. most jobs involve short processing requests on any particular resource), or of a non-homogeneous stream with a mixture of long and short processing requests. If the workload is of the first type we expect the coefficient of variation of the service time ($c = \sigma^2/\mu^2$) to be less than 1, whereas if the workload is of the second type, c is expected to be greater than 1.

We were not able to determine the value of c exactly, but we managed to approximate it when examining the performance monitor statistics.

5. Estimation of output measures

The criteria which have been used for comparison between the model and system outputs were the mean queue length at each resource and the utilisation factors. The variance of queue lengths were difficult to estimate properly in the system we used, because snapshots were taken at long time intervals (2 min).

If snapshots were to be taken at much smaller intervals of time, unacceptably high system overheads would be incurred.

5.1 Mean queue length at each resource

An output measure predicted by the model is the mean queue length at each resource when the system is in a steady state. However, as mentioned in Section 3, the performance package does not record the mean queue length as such. It only records snapshots of the queue lengths at each resource. Nevertheless, if the system has been running for some time without breakdown, i.e. at (or nearly at) steady state, the average of a number of consecutive snapshots over a period of time could be considered an estimate of the mean queue length. Though one cannot

Table 3 Performance measures

	Mean queue length				Resource utilisation			
	1 ^b	2 ^b	3 ^b	4 ^b	1	2	3	4
<i>Condition (a)*</i>								
Model predictions	9.477	0.177	1.177	0.382	0.997	0.149	0.591	0.280
System performance	9.600	0.200	1.200	0.380	0.818	0.209	0.48	0.235
Value of abs. error	0.123	0.023	0.023	0.002				
Rel. error (%)	1.3	11.5	1.9	5.3				
<i>Condition (b)*</i>								
Model predictions	9.750	0.600	5.115	0.700	0.998	0.265	0.742	0.304
System performance	9.900	1.900	5.100	1.100	0.968	0.256	0.719	0.294
Value of abs. error	0.150	1.300	0.015	0.400				
Rel. error (%)	1.5	68	0.3	36				
<i>Condition (c)*</i>								
Model predictions	16.400	0.320	1.530	0.770	0.972	0.292	0.890	0.589
System performance	15.550	0.850	1.450	0.500	0.844	0.253	0.775	0.511
Value of abs. error	0.850	0.530	0.080	0.270				
Rel. error (%)	5.5	62	5.5	54				
<i>Condition (d)*</i>								
Model predictions	9.700	0.285	4.800	0.360	0.992	0.179	0.803	0.219
System performance	10.650	1.000	4.000	0.350	0.646	0.12	0.523	0.143
Value of abs. error	0.950	0.715	0.800	0.010				
Rel. error (%)	8.9	71	20	2.9				
^a Details of the conditions are								
Condition	Time interval	No of jobs, N	$\mu_1^{-1} \times 10^4$ (min)		p_{11}			c
(a)	01.32–01.40	11	3.4		(0.0046 0.4831 0.3610 0.1513)			0.8
(b)	09.58–10.06	18	2.3		(0.0043 0.5810 0.3063 0.1084)			2
(c)	17.05–17.13	18	1.83		(0.0039 0.5234 0.3010 0.1717)			0.4
(d)	21.55–22.03	16	2.83		(0.0070 0.4861 0.4101 0.0968)			1.5

^b 1 \equiv CPU; 2 \equiv controller + 2HSD; 3 \equiv controller + 2EDS60; 4 \equiv controller + 5EDS60

guarantee this to be true, it is more likely that the queue length when the system is at steady state will not be dramatically different from its mean value over long periods of time. To ensure that the above statement is valid, the average queue length in each case under study has been taken over two (unoverlapped) consecutive periods of 10 min each (5 snapshots in this case). As expected, the two averages, in most of the cases were very close.

5.2 The average resource utilisation

The system performance data include the utilisation of the CPU, but not of the I/O resources. However, an estimate of the utilisation of each I/O resource can be worked out as follows. If u_1 is the CPU utilisation as delivered by the performance package and u_i ($i = 2, 3, \dots, M$) denotes the utilisation of each I/O resource, then the rate of input to the i th resource is $u_1 \mu_1 p_{1i}$ and the rate of output from the i th resource is $u_i \mu_i$. When the system is at steady state, the two rates should be equal; thus

$$u_i = (u_1 \mu_1) p_{1i} \frac{1}{\mu_i}$$

6. Results of the comparison process

The model prediction and the corresponding system performance statistics were examined and compared using examples of data collected at different times of day. These had different coefficients of variation c of the service times. The results are displayed in Table 3 where the device numbers 1–4 denote respectively the CPU, a controller with two HSD, a controller with two EDS60 and a controller with five EDS60.

Utilisations

The results given by the model for the resource utilisations u_i ($i = 1, 2, 3, 4$) show reasonably good agreement with those for the actual system. For example, the relative error of the CPU utilisation u_1 was as small as 3% in Condition (b) and about 15% in Condition (c); for the HSD (u_2) it was 3.5% in Condition (b) and $\sim 15.4\%$ in Condition (c).

References

- ABATE *et al.* (1968). Queueing analysis of the IBM 2314 disk storage facility, *Journal of the ACM*, Vol. 15 No. 9, pp. 577–589.
- ANDERSON JR, H. A., REISER M. and GALATI, G. L. (1973). The characterization and classification of the interactive workload for virtual memory computer system, in *Proceedings of Computer Science and Statistics: The Seventh Annual Symposium on the Interface*, Iowa State University, Ames, Iowa.
- BERNERS-LEE, C. M. (1972). Three analytic models of batch processing systems, in *BCS Conference on Computer Performance*, University of Surrey, pp. 43–52. British Computer Society, London.
- BUZEN, J. P. (1971). Analysis of system bottlenecks using a queueing network model, in *Proceedings of ACM-SIGOPS Workshop on System Performance Evaluation*, pp. 82–103.
- BUZEN, J. P. (1973). Computational algorithms for closed queueing networks with exponential servers, *Communications of the ACM*, Vol. 16 No. 9, pp. 527–531.
- CHANDY, K. M. (1972). The analysis and solutions for general queueing networks, in *Proceedings of the 6th Annual Conference on Information Sciences and Systems*, pp. 224–228.
- CHANDY, K. M., HERZOG, U. and WOO, L. (1975a). Parametric analysis of queueing networks, *IBM Journal of Research and Development*, Vol. 19 No. 1, pp. 36–42.
- CHANDY, K. M., HERZOG, U. and WOO, L. (1975b). Approximate analysis of general queueing networks, *IBM Journal of Research and Development*, Vol. 19 No. 1, pp. 43–49.
- GELENBE, E. (1975). On approximate computer system models, *Journal of the ACM*, Vol. 22 No. 2, pp. 261–269.
- GELENBE, E. and PUJOLLE, G. (1977). A diffusion model for multiple class queueing networks, Rapport de Recherche No. 242, IRIA/LABORIA—Le Chesnay (France).
- GIAMMO, T. (1976). Validation of a computer performance model of the exponential queueing network family, *Acta Informatica*, Vol. 7, pp. 137–152.
- GORDON, W. J. and NEWELL, G. F. (1967). Closed queueing systems with exponential servers, *Operations Research*, Vol. 17 No. 2, pp. 254–265.
- KOBAYASHI, H. and REISER, M. (1974). Accuracy of the diffusion approximation for some queueing systems, *IBM Journal of Research and Development*, Vol. 18 No. 2, pp. 110–124.
- MOORE, C. G. (1971). Network models for large-scale time-sharing systems, PhD Dissertation, University of Michigan, Ann Arbor.
- SALMAN, O. (1978). Queueing network models for computer system performance prediction, PhD Thesis, University of Nottingham.
- SEAMAN *et al.* (1966). An analysis of auxiliary storage activity, *IBM Systems Journal*, Vol. 5 No. 3, pp. 158–170.

Queue lengths

The model predictions of queue lengths were, as would be expected, most accurate when the queues were longer. The information on queue lengths from Conditions (a)–(d) are also shown in Table 3. It can be seen that for the CPU, for example, where the queue length was generally large in all four conditions (indicating a CPU bound system), the relative error ranged from 1.3% [Condition (a)] to a worst value of 8.9% [Condition (d)]. Similarly, with the next most heavily used device (device 3, EDS60), it ranged from 0.3% [Condition (b)] to 20% [Condition (d)]. Although the error appears larger in relative terms when queue lengths are small, the absolute value of the error remains small throughout, only exceeding the value 1 in one instance. The fact that the model predictions were generally less accurate when queue lengths were shorter is not a serious drawback, since the model correctly predicts which devices will have the longer queue lengths and which will, therefore, be significant bottlenecks affecting performance.

7. Conclusions

Validating an analytic model is not a straightforward task. The statistics typically collected from a computer system are not suitable for immediate use in validation. However, this paper has shown that by careful examination and manipulation of these statistics it is possible to extract from them the values of the input parameters required by the model (Section 2). Similarly, the two main performance measures of mean queue length and device utilisation had to be estimated from those statistics before they could be directly compared with the corresponding model predictions. All this is necessary to complete the validation process.

The validation has thus shown that computer system performance can be satisfactorily predicted using an analytic model of the central-server type.

Acknowledgements

The authors would like to express their thanks to members of The Cripps Computing Centre at the University of Nottingham for their assistance in the above work.