Optimisation of main memory size versus speed

R. L. Hittos* and D. S. Henderson†

This paper describes the application of formulae, derived from a queuing theory analysis of machine maintenance theory, to a multiprogramming computer system. Simple formulae are developed which can be used as a first approximation to optimise the size and speed of main core storage for a given job load. Some examples of the use of the formulae are given.

(Received December 1973)

1. Introduction

For many of the currently available computer mainframes core storage is available with a range of cycle speeds and the user has to decide which range of size and speed best suits his requirements. The user with a limited budget who requires an increase in throughput will have to decide whether to increase the size of his main memory and reduce the speed, or vice versa. This decision has normally to be taken either by intelligent guess-work or use of a simulation model.

In this paper we will develop a simple analytical approach to the relationship between throughput and the size and speed of core storage. The formula will be tested by use of the simulation model. We will show that it is possible to maximize throughput for a given job load and memory cost, by varying the size and speed of main memory.

2. Application of renewal theory to a multiprogramming computer system

Queueing theory has been used to analyse the repair and maintenance of machinery (Morse, 1958). In particular we are interested in formulae which have been developed to treat the situation in which K machines are serviced by a single repair crew. If it is assumed that the breakdown-time distribution, or mean time between failure distribution and the mean time to repair distribution are both exponential, formulae can be derived giving the number of machines out of commission, the utilisation of the repair crew, etc.

We now consider the above situation to be analogous to that of a multiprogramming computer system in which a single CPU serves a number of program tasks with exponentially distributed service time. The time for which the machines are operating we consider analogous to the time in which the task's input/output requests are being serviced, and the repair time is of course analogous to the time for which the task uses the CPU. To obtain tractable formulae we assume that this time is also exponentially distributed.

Consider K program tasks in core, with mean I/O time T_A and mean CPU time T_s. Applying the machine repair formulae we have: The mean fraction of the time the CPU is busy is given by:

$$1 - Po = 1 - (x^K e^{-x}/K!\, E_K(x)) \qquad (1)$$

where

$$x = T_A/T_s$$

and

$$e^x E_K(x) = \sum_{n=0}^{K} (x^n/n!) \qquad (2)$$

Po is the probability that no tasks are in the CPU queue, therefore $1 - Po$ is the CPU utilisation.

We assume further that system throughput is proportional to CPU utilisation and proportional to main memory speed, which is inversely proportional to T_s, the mean CPU service time.

We justify the above assumption by noting that the number of memory references made by any program is constant, and if CPU utilisation increases, the total number of memory references made by the CPU per interval of time, and therefore the progress rate of tasks through the system, will increase. We therefore say that

$$\text{Throughput} = K_t(1 - Po)/(T_s/T_A) \qquad (3)$$

where K_t is constant of proportionality.

We assume that T_A, the average I/O time, is dependent only on the speed of the I/O devices and not upon main memory speed, and that it will therefore remain constant.

3. Relationship between memory cost and memory size and speed

We assume that, for a given memory speed, price is proportional to the size of the core storage. For a given job mix, if the average task size is m, the average memory size required will be $K.m$, if K is the average number of tasks which will multiprogram. The price will therefore be proportional to K.

To illustrate the relationships graphically it is necessary to assume some relationship between memory cost and memory speeds.

Grosch's law states that computing power is proportional to the square of the price. We can make the assumption that computing power is proportional to memory speed, which will be true providing that only a small proportion of the computing is performed in registers of which speed is independent of main memory speed. Grosch's law then implies that memory speed is proportional to the square of the price.

Alternately, we can simply assume that memory speed is proportional to price.

We can then obtain some relationship between memory cost G and memory speed of the form:

$$G = k_m K/(T_s/T_A)^n \qquad (4)$$

where $n = \frac{1}{2}$ for the Grosch's law case, $n = 1$ for the linear case. k_m is some constant of proportionality.

4. Relationship between memory cost and throughput

As we can easily see from Equation 4, for a fixed memory cost we have a discrete range of memory speeds and memory sizes, corresponding to the average no. of tasks in core, K, having values 1, 2, 3, 4, etc.

Using Equations 3 and 4 we can calculate the value of the system throughput for each combination of K and T_s/T_a which keep the memory cost constant. **Fig. 1** shows graphs of throughput versus T_s/T_A for different prices, and different values of n.

As we are only interested in the shape of the graphs, we plot only the 'throughput factor', $(1 - Po)/(T_s/T_A)$ against T_s/T_A, keeping the 'price factor' $K/(T_s/T_A)^n$ constant.

*Control Data South Africa, P.O. Box 78105, Sandton 2146, South Africa.
†Department of Applied Mathematics, University of the Witwatersrand, Johannesburg, South Africa.

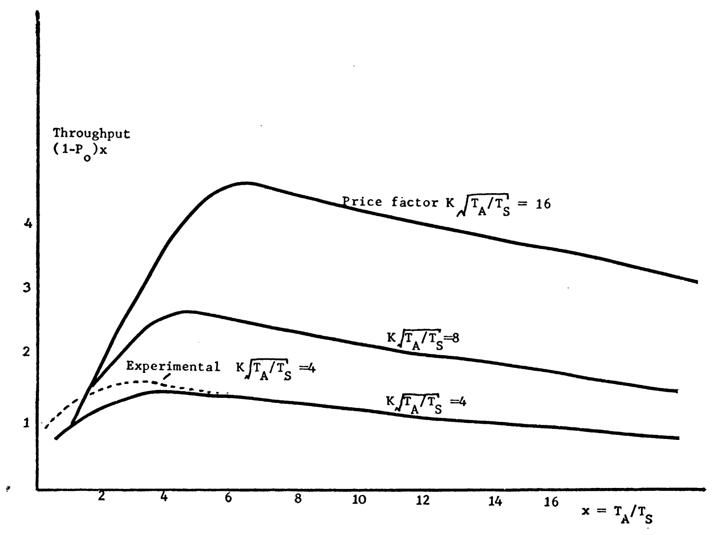

Fig. 1(a) Throughput vs. T_A/T_S for constant memory cost and quadratic price-memory speed relationship. (Price $\alpha T_S/T_A$.)

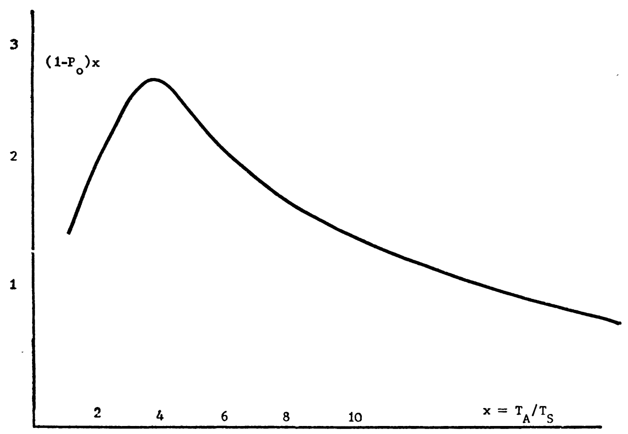

Fig. 1(b) Linear price-speed relationship $K(T_A/T_S) = 16$

From the figure we see that correct choice of memory speed and size can maximise the throughput of a computer system while the cost is kept constant.

The relationship was checked experimentally using a simulation model of a multiprogramming operating system and a job mix of which all tasks had the same size. The price factor for the experiment was taken as 4, with $n = \frac{1}{2}$. The experiments confirm the existence of a maximum value for throughput, and agree reasonably well with the theoretical results. (Fig. 1(a)).

5. Conclusions

In deriving the above formulae certain assumptions were made which limit the validity of the formulae. In deriving formula (1) it was assumed that the CPU services times, T_s, and the arrival rate at the CPU queue were both exponentially distributed. This assumption can safely be made only if a sufficiently large number of programs share the CPU, and provided that no significant queueing occurs for the I/O channels. As soon as I/O queuing becomes significant the problem has to be approached by analysis of the channel and I/O device queues.

Another simplification caused by our use of machine maintenance theory is the assumption that the number of tasks which multiprogram remain constant. This implies a fixed number of partitions. Where a varying number of tasks are allowed to multiprogram, the average number of tasks can be used as a first approximation. It would be possible to extend the theory to cover the deviation of k from the mean; however it is doubtful whether the possible improvement in the formula is justifiable in view of the previous simplifications.

The formulae in this paper are therefore presented to be used as a first approximation for an answer to a problem which is usually tackled by intuition alone. A few examples will illustrate the way in which the formulae can be used.

6. Examples

Example 1

At one time the costs of an IBM 360/50 with (a) 256K and 2μ seconds storage and (b) 1024K and 8μ seconds storage were almost identical. The operating system was planned to occupy 64K of storage and a typical job occupied a partition of 192K. It's ratio T_A/T_s of I/O to CPU time was four in the fast storage and thus one in the slow storage. The comparative situation may be summarised as:

Case	T_A/T_s	*No. of partitions*	*CPU utilisation*	*Throughput*
a	4	1	0·2	0·8
b	1	5	0·997	0·997

For these operating parameters the choice of case (b) in preference to (a) results in an almost 25 per cent increase in throughput.

At the University of the Witwatersrand, choice (b) was initially installed, the decision being largely intuitive, since the characteristics of the future job-stream were only known in broad outline. Subsequently, by fortuitous financial and historical circumstances choices (a) and (b) were installed side by side in the same machine room. In spite of the gross simplification of the description of the job-stream characteristics, the relative throughputs of the two machines in a typical university environment were found to bear substantially the above relationship to each other.

Example 2

(a) In a certain installation the cost of the hardware (peripherals, CPU and storage required by the operating system) is normalised to one unit. On this scale the cost per user storage partition is 0·125. For the anticipated job-stream the ratio T_A/T_s is estimated at 4. How many user storage partitions should be provided (constant job size) to maximise the throughput: cost ratio?

$$c = 1 + K/8; x = 4$$

No. of partitions K	*Cost C*	*Throughput $(1 - Po)x$*	*Throughput/cost*
1	1·125	0·8	0·72
2	1·250	1·54	1·23
3	1·375	2·2	1·60
4	1·500	2·76	1·84
5	1·625	3·2	1·97
6	1·750	3·55	2·02
7	2·000	3·75	1·86

six storage partitions is therefore the optimum number.

(b) The exercise of 2(a) is repeated with storage four times slower and with a unit cost of 0·06. As one would intuitively expect the optimum number of partitions is lower.

No. of partitions	*Cost*	*Throughput*	*Throughput/cost*
1	1·06	0·5	0·47
2	1·12	0·8	0·72
3	1·18	0·94	0·80
4	1·24	0·98	0·79
5	1·30	0·99	0·76

Example 3

The machine of example 2(a) with one user partition of real storage is run under a virtual storage operating system. As the number of virtual partitions goes up the amount of real time wasted through paging has been observed as reported below. The effect of this wasted time is discounted by regarding the machine as having all real storage with a slower access time and consequent reduction of the ratio $x = T_A/T_s$. For a constant unit cost of 1·125 the situation is summarised below.

K	*x*	*Wasted paging fraction*	$1 - Po$	*Throughput*	*Throughput/ cost*
1	4	0	0·2	0·8	0·72
2	3	0·25	0·47	1·41	1·25
3	2	0·50	0·79	1·58	1·4
4	1	0·75	0·98	0·98	0·87

Three virtual partitions would appear to be the best operating strategy.

The desirability of spending a further 0·125 cost units to acquire a second real partition is to be investigated. It is assumed, perhaps not entirely justifiably, that the same paging characteristics will apply to the same over-commitment ratios. The position may be summarised:

K	*x*	*Wasted paging fraction*	$1 - Po$	*Throughput*	*Throughput/ cost*
2	4	0	0·38	1·54	1·23
4	3	0·25	0·79	2·37	1·93
6	2	0·50	0·99	1·98	1·58
8	1	0·75	1·0	1·0	0·8

The optimum operating point is four virtual partitions. It is interesting to compare the four virtual partitions of this example with the four real partitions of Example 2(*a*). The throughput in the virtual case is almost 90 per cent of the four real partitions, and the performance/cost ratio is some 3 per cent better. (In practice this small improvement would be swallowed up in increased systems overheads).

Reference

MORSE, P. M. (1958). *Queues, Inventories and Maintenance*, John Wiley and Sons, pp. 167.

Book review

The Chebyshev Polynomials by Theodore J. Rivlin, 1974; 186 pages. (*John Wiley*, £8·60)

Chapter 1 (55 pages) of this book lists some elementary properties of the Chebyshev polynomials, and then considers Lagrangian interpolation with some relevant nodes, showing that no choice gives convergence, as the degree of the approximating polynomial increases, for every continuous function. Hermite interpolation at the Chebyshev zeros does achieve this, and Lagrange-Chebyshev interpolation at least converges in the mean for every continuous function. Further topics include orthogonal polynomials, the differential equations, recurrence relations and generating functions for the Chebyshev polynomials, and numerical integration and the Gauss-Chebyshev formula.

Chapter 2 (68 pages) discusses convex sets and the characterisation of the best approximation in the uniform norm, a variant in terms of extremal signatures, the Chebyshev condition for best polynomial approximation, the Haar criteria and uniqueness, and the minimax theory for an interval of the real axis. The second part of the chapter examines classes of linear functionals for which Chebyshev polynomials are extremal elements particularly in the space of polynomials, and discusses growth outside the interval, a generalisation of the Lanczos τ method for approximating e^x, and methods for determining bounds for the derivatives of a polynomial.

Chapter 3 (36 pages) treats economisation, the evaluation of a finite Chebyshev series, Chebyshev expansions, absolute and uniform convergence of Chebyshev series, the relation between least squares and uniform errors of truncated Chebyshev expansions, bounds for the errors in terms of the coefficients and for the sizes of the coefficients, their computation by various quadrature rules, optimal properties of Chebyshev expansions, and the ellipse of convergence of Chebyshev series.

Chapter 4 (7 pages) discusses briefly the identity $T_m(T_n) = T_n(T_m) = T_{mn}$, the properties of permuting and commuting polynomials and the ergodic and mixing properties of the mapping T_n^{-1}, the sequence of mappings $T_1^{-1}, T_2^{-1}, \ldots, T_n^{-1}$, and the iterates T_n^{-k}, the k-fold composition of T_n^{-1}.

This book is everywhere dense with mathematical information, analysis and over 200 relevant exercises. It aims to give the mathematical student a taste of the excitement stimulated by the Chebyshev polynomial in various areas of analysis, and to serve as leisure reading for a broader mathematical community. In both these respects it will clearly succeed. There is, however, little here for the practical man, scientist or applied numerical analyst, and one would perhaps have liked something on the 'applications of Chebyshev polynomials in kinematics', one of the topics (with number-theoretic aspects) excluded deliberately 'through ignorance'. There is certainly nothing ignorant about what has been included!

L. Fox (Oxford)