A note on the influence of program loading on the page fault rate

Michel Parent, Dominique Potier

IRIA

laboria

laboratoire de recherche
en informatique
et automatique

A NOTE ON THE INFLUENCE OF PROGRAM LOADING ON THE PAGE FAULT RATE

Michel Dominique

M. Parent, D. Potier

Rapport de Recherche N° 159
Mars 1976

Institut de Recherche
d'Informatique
et d'Automatique

Domaine de Voluceau
Rocquencourt
B.P. 5 78150 - Le Chesnay
France
Tél.: 954 90 20

E R R A T U M

Les pages 5 et 6 doivent être inversées.

Pages 5 and 6 have to be interchanged.

A NOTE ON THE INFLUENCE OF PROGRAM LOADING ON THE PAGE FAULT RATE

M. Parent, D. Potier
IRIA/LABORIA

Résumé :

Dans la modélisation des systèmes informatiques à mémoire virtuelle, la fonction de durée de vie qui exprime le temps moyen entre fautes de pages pour un programme donné en fonction de l'espace mémoire alloué joue un rôle important. Cet article présente un modèle qui permet d'étudier l'influence sur la durée de vie des défauts de page initiaux nécessaires pour remplir l'espace mémoire alloué et montre que cette influence peut être importante losque le temps moyen de résidence du programme en mémoire principale est petit.

Abstract :

In virtual memory computer system modeling the life time function (LTF) which expresses the mean virtual time between page faults of a given program as a function of memory space allocated to it, plays an important role. This paper presents a model to study the influence of the initial page faults needed to fill the memory allocation on the LTF and it is shown that this influence is far from negligeable when the mean residency time of the program in main memory is small.

A NOTE ON THE INFLUENCE OF PROGRAM LOADING ON THE PAGE FAULT RATE

I - INTRODUCTION

Many models of paged virtual memory systems based on queueing networks have been developped recently in order to predict their performances [2, 3, 10, 11, 20]. The main difficulty encountered during the definition of these models is the characterization of program behavior with respect to their memory requirements [13]. The framework of queueing network representation implies that memory requirements have to be expressed in some way or other as demands on servers : in other words, space requirements have to be transformed into time requirements.

In the context of page on demand systems [12] this transformation can be done by the so-called "life time function" (LTF) which relates for a given program, the mean time between two consecutive page faults to the number of page frames allocated to the program. The LTF of a program is defined and measured in the virtual time of the program for a given page replacement algorithm (PRA) and a given page size [8]. If a program execution trace is available, the LTF of the program can be obtained easily for a large class of PRA using the technique of stack processing [19, 6]. Hence, for a given PRA and a given page size, the LTF can be considered as an intrinsic model of the program paging activity.

Examples of experimental LTF's of a FORTRAN compiler for various page sizes and an LRU PRA are presented on figure 1. The leveling of all the curves is du to the fact that a fixed set of pages has to be loaded whatever the allocation of main memory is and to the fact that program execution time is finite. This phenomenon can be accentuated if a program is interrupted before its completion and loses pages during these interruptions. In particular, for very short transactions, the swapping out of programs can increase considerably the page fault rate if the pages are reloaded on a page-on-demand basis and this may become the dominant factor in the paging behavior of the program [15]. This comes from the fact that programs tend to demand pages at a very rapid rate until they have acquired a sufficiency of pages (which can be thought of a the "working set") [16]. An example of a model where the influence of the initial page faults has been considered can be found in [17].

Since this phenomenon is not intrinsic of the program itself it is important not to include it in the LTF as it is obtained with stack processing for example. The best way to do this is to consider that the mean time between page faults is function of the number of pages actually loaded in main memory and not of the number of page frames allocated. The LTF defined this way can be obtained as previously if we discard the initial page faults needed to fill the space allocation.

To a given LTF defined as above, we can associate a modified life-time function (MLTF) which represents the run-time environment mean time between page faults as a function of allocated core space when the interruptions and the swapping policy are taken into account. We present in this paper a model to derive the MLTF from the LTF for various swapping policies. The results are obtained for a simple model of the LTF and illustrate how the LTF can be distorded by factors other than intrinsic program behaviour. These effects are demonstrated on a global model of a virtual memory multiprogrammed machine, and the results are used to provide an upper bound of the performance improvement which can be achieved by preloading the program before each activation.

II - THE MODEL OF PROGRAM BEHAVIOR

In the context of multiprogrammed computer systems, the execution of a program consists of a sequence of CPU intervals and interruptions due to I/O requests, page-faults, end of time-slice, or end of execution. Depending on the nature of these interruptions and on the memory allocation policy, when a process starts a new CPU interval, it does or does not still possess the set of pages loaded in main memory immediately before the last interruption.

In this study, we shall only be concerned with two types of interruptions : page fault interruptions which increase the number of pages loaded in main memory by one, and program interruptions such that during the time interval elapsed between the interruption and the reactivation of the program, some pages may have been swapped out. We will also make the following assumptions :

H1 : The CPU intervals between program interruptions as defined above are independent, identical exponentially distributed random variables with mean T.

H2 : The maximum number M of page frames allocated to a program is fixed and pages are loaded in main memory on a page on demand basis .

H3 : The intervals of time between two consecutive page faults of a program are independent, identical exponentially distributed random variables with mean q_i when i pages of the program are loaded in main memory (q_i represents the LTF of the program as it has been defined in the introduction).

H4 : The transition between the number i of pages of the program loaded in memory when the program is interrupted (program interruption) and the number j of pages loaded in memory when it starts a new CPU interval is described by a first order Markov Chain with transition matrix (α_{ij}).

The main assumption (H3) used in this modeling is to consider that the time between page faults is a negative exponential random variable which depends only on the actual number of pages in main memory (intrinsic life-time function). Experiments conducted on real trace data [9] show that this assumption is not unfounded but the random variable has an hyperexponential distribution. In any case assumption H3 (the other assumptions are much weaker) has been proved to give satisfactory results in a model of virtual memory system [7, 14] and we can infer from this that it is justified.

Under assumptions H1 - H4, the execution of a program in virtual time (program running time) consists of a number of CPU intervals separated by program interruptions and page fault interruptions (the length of these interruptions is not taken into account in virtual time). If a program has i pages loaded in main memory when it starts a CPU interval, at the next program interruption, the program will have j pages in main memory with

$$j = \min(i+k,\ M)$$

where k is the number of page faults which occured during the time considered. At the beginning of the next CPU interval, the program will **start** with ℓ pages ($\ell \leq j$) with probability $\alpha_{j\ell}$.

Let us define the state X(t) of a program at a given instant t of its virtual time as the number of its pages it possesses in main memory. The set of states E is then

$$E = \{1,\ 2, \dots, M\}$$

and the state transition diagram is represented on figure 2. With the assumptions H1, H2, H3, H4, X(t) is a semi-Markov process which we can now analyze. Let $P = (p_{i,j})$ be its transition matrix. Transitions are caused by two different events : page fault and program interruption. From the exponential assumptions H1 and H3, the probability σ_i that the next event to occur is a page fault when the program is in state i, can be expressed as :

$$\sigma_i = \frac{1/q_i}{1/q_i + 1/T} \qquad (1)$$

and the mean virtual time e_i in state i is given by

$$e_i = \frac{1}{1/q_i + 1/T} \qquad (2)$$

With probability $1 - \sigma_i$ the program will be interrupted in state i, and will be reactivated in state j with probability α_{ij} from H4. Hence, we have

From equation (7) we can define the memory utilization ratio m/M which represents the mean fraction of pages frames actually used during the execution of a program, and equation (8) gives the MLTF r_M derived from the LTF q_i.

In the appendix, closed form solutions of equation (4) are given for various transition matrices α_{ij}.

III - NUMERICAL RESULTS

The previous analysis provides theoretical results for the evaluation of the effect of program loading in main memory on a demand paging basis. The two main factors considered here are the mean time T between program interruptions and the number of pages which are in main memory after the processing of the interruption. For the numerical results, it was assumed that the life-time function of programs, that is, the mean virtual time e between two page faults when a program has i pages loaded in main memory, is given by the Belady model [5] :

$$e = \alpha\, i^k$$

where α and k depend on program characteristics and on the processor speed. This model has been validated using real trace data [9] and the results obtained from these experiments show values of k ranging from 1.5 to 4.

The MLTF and the memory utilization ratio have been plotted (Fig. 4, 5) according to the number of page frames allocated to a program for various values of T and k (to eliminate the influence of α, all times have been divided by α). The four models solved in the appendix have been used to describe the number of pages remaining in memory after a program interruption which occurs when i pages are loaded :

Model A :

After the interruption, the program finds a single page in memory (systematic swap-out).

Model B :

The program may find any number j, $(1 \leq j \leq i)$ of pages after the interruption with equal probabilities (random page losses).

$$p_{i,j} = \begin{cases} \sigma_i & i=1,\ldots M-1 \;;\; j = i+1 \\ (1-\sigma_i)\alpha_{ij} & i=1,\ldots M-1 \;;\; j = 1,\ldots i \\ (1-\sigma_M)\alpha_{Mj} & i=M \;;\; j=1,\ldots M-1 \\ (1-\sigma_M)\alpha_{MM}+\sigma_M & i=M \;;\; j=M \\ 0 & \text{otherwise} \end{cases} \qquad (3)$$

Let $\Pi = (\Pi_1,\ldots,\Pi_M)$ be the vector of steady state probabilities defined by

$$\Pi = \Pi P \qquad (4)$$

$$\sum_{i=1}^{M} \Pi_i = 1 \qquad (5)$$

Π_i represent the steady state probability that the program is in state i when a transition takes place. The vector of steady state probabilities $\Gamma = (\gamma_1,\ldots,\gamma_M)$, where γ_i is the steady state probability that $X(t) = i$, is obtained from Π and e_i, $i=1,\ldots M$ by

$$\gamma_i = \frac{\Pi_i e_i}{\sum_{j=1}^{M} \Pi_j e_j} \quad , i = 1,\ldots M. \qquad (6)$$

Hence in the virtual time of the program, the actual mean number m of pages loaded in main memory is given by

$$m = \sum_{i=1}^{M} i\,\gamma_i, \qquad (7)$$

and the actual mean time r_M between page faults when a maximum number M of page frames are allocated is

$$r_M = \sum_{i=1}^{M} q_i\,\gamma_i\;, \qquad (8)$$

with $m \leq M$

$r_M \leq q_M$

Model C :

The program may find all its i pages back with probability $(1-\beta)$ or one single page with probability β after the interruption (swap-out with probability β). Two values have been selected for β : $\beta = 0.2$ and $\beta = 0.8$.

Model D :

With this model, the page loss process is studied in more detail and the influence of the interruption time is considered (small interruption times lead to small page losses).

For all models and with the selected parameters for T and k, the MLTF flattens out while the LTF is strictly convex. Hence, depending on the parameters for a given memory size allocated to a program, large differences between the LTF and the various MLTF's can be observed, and these differences increase with k as it can be seen from fig. 4 and cannot be disregarded for typical values of k, i.e. $k \doteq 3$.

For the memory utilization, (fig. 5) it can also be observed that for all models, the effective use decreases rapidly (expecially for large values of k) with the number of pages allocated to a program. In some systems [1] this problem is minimized by overallocating memory space, but it is quite difficult to estimate the percentage which can be reasonably overallocated. It has to be mentionned here that these utilizations have been computed for one program with respect to its virtual time. To compute real memory utilization, the multiprogramming effect has to be considered in order to take into account the waiting times after a page fault. A two or three server model may be used to do this if we distribute programs into classes according to the number of their pages which are loaded. Although the analytical solution of such a model is known [4] numerical results would be difficult to obtain due to the large number of classes (m classes if m is the maximum number of pages allocated to a program).

The MLTF's computed for model A (systematic swap-out after a program interruption) have been used to evaluate the influence of the mean time T between program interruptions on CPU utilization in a multiprogramming system. The computer system was modelled by a central server queueing network as described in [7]. The three servers are the CPU, the paging drum and a disk file system (see fig. 6) and it is assumed that program interruptions due to a file access do not cause a swap-out. Overheads are taken into account for each kind of interruption of CPU time (page fault, file I/O, transaction end). Other applications of this model can be found in [14] and [16] where it was used to point out conditions of optimality for multiprogramming.

Figure 7 displays the CPU utilization versus the multiprogramming level for various values of T and k. For a given value of the locality k, it can be seen that small interaction times may cause a sensible reduction of the CPU utilization an hence of the system throughput especially in the optimum range of the system ($k = 2$, $n = 4$ in the exemple). One can also notice on this example that the optimum multiprogramming level (for identical programs) was independent of the time T. Similar results were obtained for other configurations and also with a simpler two servers model.

This example illustrates the worst case degradation since we suppose that the entire program is swapped out at each interruption and is afterwards reloaded on a page on demand basis starting with one single page. Hence, if interrupted programs can expect to find more than one page after the processing of an interruption, the performance of the system will increase up to the maximum value obtained with the original LTF which was closely approximated in the example by the curve $T = 5000$ msec. This is the idea behind the technique of prepaging where, after each swap-out, the monitor reloads more than one page at the end of the interruption. Depending on the system, the entire or part of the core allocation is loaded or the working set as it was at the time of interruption is loaded before the program is allowed to run. The analysis presented here provides an estimation of the maximum gain which can be expected from this technique.

APPENDIX

Solution of the model

We consider three different matrices α_{ij}, namely α^A_{ij}, α^B_{ij}, α^C_{ij}. In model A, we assume that α_{ij} is defined as follows

$$\alpha^A_{ij} = \begin{cases} 1 & \text{if } j = 1 \\ 0 & \text{if } j > 1 \end{cases} \qquad i = 1,\ldots M.$$

Model A represents the case where a program has all its pages swapped out when it is interrupted, and is reactivated with a single page.

In model B, we assume that when a process is interrupted in state i, it will be reactivated in any of the states 1, 2,... i with equal probaiblity 1/i. We therefore have

$$\alpha^B_{ij} = \begin{cases} 1/i & j = 1,\ldots i \\ 0 & j > i \end{cases} \qquad i = 1,\ldots M\cdot$$

In model C, a program interrupted in state i is assumed to be reactivated in state 1 with probability β and in state i with probability $1-\beta$. We then have

$$\alpha^C_{ij} = \begin{cases} \beta & \text{if } j = 1 \\ 1-\beta & \text{if } j = i \\ 0 & \text{Otherwise} \end{cases} \qquad i = 1,\ldots M.$$

Model D is described in detail at the end of the appendix.

Solution of model A

The state transition diagram is represented in figure 3. From the definition of α^A_{ij}, we have

$$p_{ij} = \begin{cases} \sigma_i & i = 1,\ldots M-1\ ;\ j = i+1, \\ 1-\sigma_i & i = 1,\ldots M\ ;\ j = 1, \\ \sigma_M & i = M\ ;\ j = M\ , \\ 0 & \text{otherwise} \end{cases}$$

and equation (4) can be expressed :

$$
\begin{aligned}
\Pi_1 &= \sum_{i=1}^{M} (1 - \sigma_i)\, \Pi_i \,, \\
\Pi_2 &= \sigma_1\, \Pi_1 \,, \\
&\vdots \\
\Pi_i &= \sigma_{i-1}\, \Pi_{i-1} \,, \\
&\vdots \\
\Pi_M &= \sigma_{M-1}\, \Pi_{M-1} + \sigma_M\, \Pi_M .
\end{aligned}
$$

It follows immediately :

$$\Pi_i = \Pi_1 \prod_{j=1}^{i-1} \sigma_j \,, \qquad i = 2, M-1$$

$$\Pi_M = \frac{\Pi_1}{1 - \sigma_M} \prod_{j=1}^{M-1} \sigma_j \,.$$

Solution of model B

We have

$$
p_{i,j} = \begin{cases}
\sigma_i & , \quad i = 1, \ldots M-1, \ j = i+1 \\
\dfrac{1 - \sigma_i}{i} & , \quad i = 1, \ldots M-1, \ j = 1, \ldots i \\
\dfrac{1 - \sigma_M}{M} & , \quad i = M, \ j = 1, \ldots M-1 \\
\sigma_M + \dfrac{1 - \sigma_M}{M} & , \quad i = M, \ j = M \\
0 & \quad \text{otherwise}
\end{cases}
$$

and equations (4) can be written :

$$\Pi_1 = \sum_{j=1}^{M} \left(\frac{1 - \sigma_j}{j}\right) \Pi_j$$

$$\Pi_2 = \sigma_1\, \Pi_1 + \sum_{j=2}^{M} \left(\frac{1 - \sigma_j}{j}\right) \Pi_j$$

....

$$\Pi_i = \sigma_{i-1}\,\Pi_{i-1} + \sum_{j=i}^{M} \left(\frac{1-\sigma_j}{j}\right) \Pi_j$$

....

$$\Pi_M = \sigma_{M-1}\,\Pi_{M-1} + \frac{1-\sigma_M}{M}\,\Pi_M + \sigma_M\,\Pi_M$$

In order to solve this system of equations, we form the differences $\Pi_i - \Pi_{i-1}$, $i = 2, M$. We then have :

$$\Pi_i - \Pi_{i-1} = \sigma_{i-1}\,\Pi_{i-1} - \sigma_{i-2}\,\Pi_{i-2} - \frac{1-\sigma_{i-1}}{i-1}\,\Pi_{i-1} \quad , \quad i = 2, M-1,$$

which turns into

$$\left(\Pi_i - \frac{i}{i-1}\,\sigma_{i-1}\,\Pi_{i-1}\right) = \frac{i-2}{i-1}\left[\Pi_{i-1} - \frac{i-1}{i-2}\,\sigma_{i-2}\,\Pi_{i-2}\right] .$$

From this equation, the solution is simply obtained as :

$$\Pi_i = i\,\Pi_1 \prod_{j=1}^{i-1} \sigma_j$$

$$\Pi_M = M\,\frac{\Pi_1}{1-\sigma_M} \prod_{j=1}^{M-1} \sigma_j$$

Solution of model C

We have :

$$p_{ij} = \begin{cases} 1 - \sigma_1 & i = 1 \; ; \; j = 1 \\ \sigma_i & i = 1,\ldots M-1 \; ; \; j = i+1 \\ (1 - \sigma_i)\beta & i = 2,\ldots M-1 \; ; \; j = 1 \\ (1 - \sigma_i)(1-\beta) & i = 2,\ldots M-1 \; ; \; j = i \\ (1 - \sigma_M)(1-\beta) + \sigma_M & i = M \; ; \; j = M \\ 0 & \text{otherwise} \end{cases}$$

and equation (4) writes :

$$\Pi_1 = (1 - \sigma_1)\,\Pi_1 + \sum_{i=2}^{M} (1 - \sigma_i)\beta\,\Pi_i$$

$$\Pi_2 = \sigma_1\,\Pi_1 + (1 - \sigma_2)(1 - \beta)\,\Pi_2$$

....

$$\Pi_i = \sigma_{i-1}\,\Pi_{i-1} + (1 - \sigma_i)(1-\beta)\Pi_i$$

....

$$\Pi_M = \sigma_{M-1}\,\Pi_{M-1} + (1 - \sigma_M)(1-\beta)\,\Pi_M + \sigma_M\,\Pi_M$$

We obtain :

$$\Pi_i = \Pi_1 \prod_{j=1}^{i-1} \frac{\sigma_j}{(\sigma_{j+1}+\beta) - \beta\,\sigma_{j+1}}$$

$$\Pi_M = \frac{\Pi_1}{1 - \sigma_M} \prod_{j=1}^{M-1} \frac{\sigma_j}{(\sigma_{j+1} + \beta) - \beta\,\sigma_{j+1}}$$

In the three models, Π_1 is computed from equation (5) and the γ_i can be directly obtained by using equation (6).

Solution of Model D

Model D is more complex because we look in detail into the page loss process (in real time and independently of the program paging activity). We suppose that the program starts its interruption with i pages in memory and we want to find the probabilities α_{ij} for it to end its interruption with j pages $(j \leq i)$.

We will now suppose that the program looses pages one by one during the interruption. If the interruption time and the time until the next page loss when the program has j pages in memory are both exponentially distributed random variables (with means I and r_j), then the probability to loose one more page at any stage of the process is given by :

$$\delta_j = \frac{1/r_j}{1/r_j + 1/I}$$

and the matrix (α_{ij}) is obtained simply :

$$\begin{aligned}
\alpha_{i,i} &= 1 - \delta_i \\
\alpha_{i,i-1} &= \delta_i(1-\delta_{i-1}) \\
&\vdots \\
\alpha_{i,j} &= \delta_i\ \delta_{i-1}\ \ldots\ \delta_{j+1}(1-\delta_j) \\
&\vdots \\
\alpha_{i,1} &= \delta_i\ \delta_{i-1}\ \ldots\ \delta_2
\end{aligned}$$

Now, in order to compute r_i, we can assume that the page stealing rate is constant and equally distributed among a fixed number of "eligible pages". This leads to the formula :

$$r_i = \frac{\beta}{i}$$

where β can be thought of as the mean time between two pages steals for a given eligible page frame.

This leads to :

$$\delta_i = \frac{i}{i+\gamma}$$

where

$$\gamma = \frac{\beta}{I}$$

A simple closed form solution of equation (4) was not found with this model, but the numerical solution is easily obtainable starting from

Π_M through the following recurrence formula :

$$\Pi_{j-1} = \frac{1}{\sigma_{j-1}} \left[\Pi_j - \sum_{i=j}^{M} (1 - \sigma_i) \, \alpha_{ij} \, \Pi_i \right]$$

Π_M is obtained from equation (5) and the γ_i are again obtained with equation (6).

REFERENCES

[1] ADAMS J.C., MILLARD G.E. - "Performance Measurements on the Edinburgh Multi-Access System" - Proceedings of ACM - ICS 75 - Antibes, France - June 1975.

[2] AVI-ITZHAK B., HEYMAN D.P. - "Approximate Queueing Models for Multiprogramming Computer Systems" - Operation Research vol. 21, n° 6 - Nov.-Dec. 1973.

[3] BARD Y. - "An Analytic Model of CP-67 VM/370" - Proceedings of ACM-SIGARCH-SIGOPS Workshop on Virtual Computer Systems - Cambridge, Mas. 1973.

[4] BASKETT F., CHANDY K.M., MUNTZ R.R., PALACIOS F.G. - "Open, Closed and Mixed Networks of Queues with Different Classes of Customers" - Journal of ACM vol. 22, n° 2 - April 1975.

[5] BELADY L.A., KUEHNER C.J. - "Dynamic Space Sharing in Computer Systems" - Comm. of ACM vol. 12, n° 5 - May 1969.

[6] BENNETT B.T., KRUSKAL V.J. - "LRU Stack Processing" - IBM Jour. of Res. and Develop. - July 1975.

[7] BRANDWAJN A., BUZEN J., GELENBE E., POTIER D. - "A Model of Performance for Virtual Memory Systems" - ACM-SIGMETRICS Symp. - Montreal, Canada - 1974.

[8] BURGEVIN P., LENFANT J. - "Empirical Data on Program Behavior" - Proceedings of ACM - ICS 75 - Antibes, France - June 1975.

[9] BURGEVIN P., LEROUDIER J. - "Measurement Analysis of Program Behavior" - Rapport Laboria - IRIA - France - March 1976.

[10] BUZEN J.P. - "Queueing Network Models of Multiprogramming" - Ph. D. Thesis - Harvard University - Cambridge, Mas. - 1971.

[11] CHANG W. - "Sequential Servers Queues for Computer Communication System Analysis" - IMB Jour. of Res. and Develop. - Sept. 1975.

[12] DENNING P.J. - "Virtual Memory" - Computing Surveys - Sept. 1970.

[13] DENNING P.J. - "On Modeling Program Behavoir" - Spring Joint Computer Conference - 1972.

[14] DENNING P.J., KAHN K.C. - "An L=S criterion for optimal multiprogramming" - SIGMETRICS Symposium - Cambridge - March 1976.

[15] EASTON M.P., FAGIN R. - "Cold start vs warm start miss ratios and multiprogramming performance" - IBM Yorktown Research Lab report RC 5715 - November 1975.

[16] FINE G.H., JACKSON C.W., MC ISAAC P.V. - "Dynamic program behaviour under paging" - Proceedings ACM National Meeting - 1966.

[17] GELENBE E., TIBERIO P., BOEKHORST J.C.A. - "Page size in demand paging systems" - Acta Informatica - N° 3 - 1973.

[18] LEROUDIER J., POTIER D. - "Principles of optimality for multiprogramming" - SIGMETRICS Symposium Cambridge - March 1976.

[19] MATTSON R.L., GECSEI J., SLUTZ D.R., TRAIGER I.L. - "Evaluation Techniques for Storage Hierarchies" - IBM Syst. Jour. n° 2 - 1970.

[20] SHEDLER G.S. - "A Queueing Model of a Multiprogrammed Computer with a Two Level Storage System" - Comm. if ACM vol. 16, n° 1 - Jan. 1976.

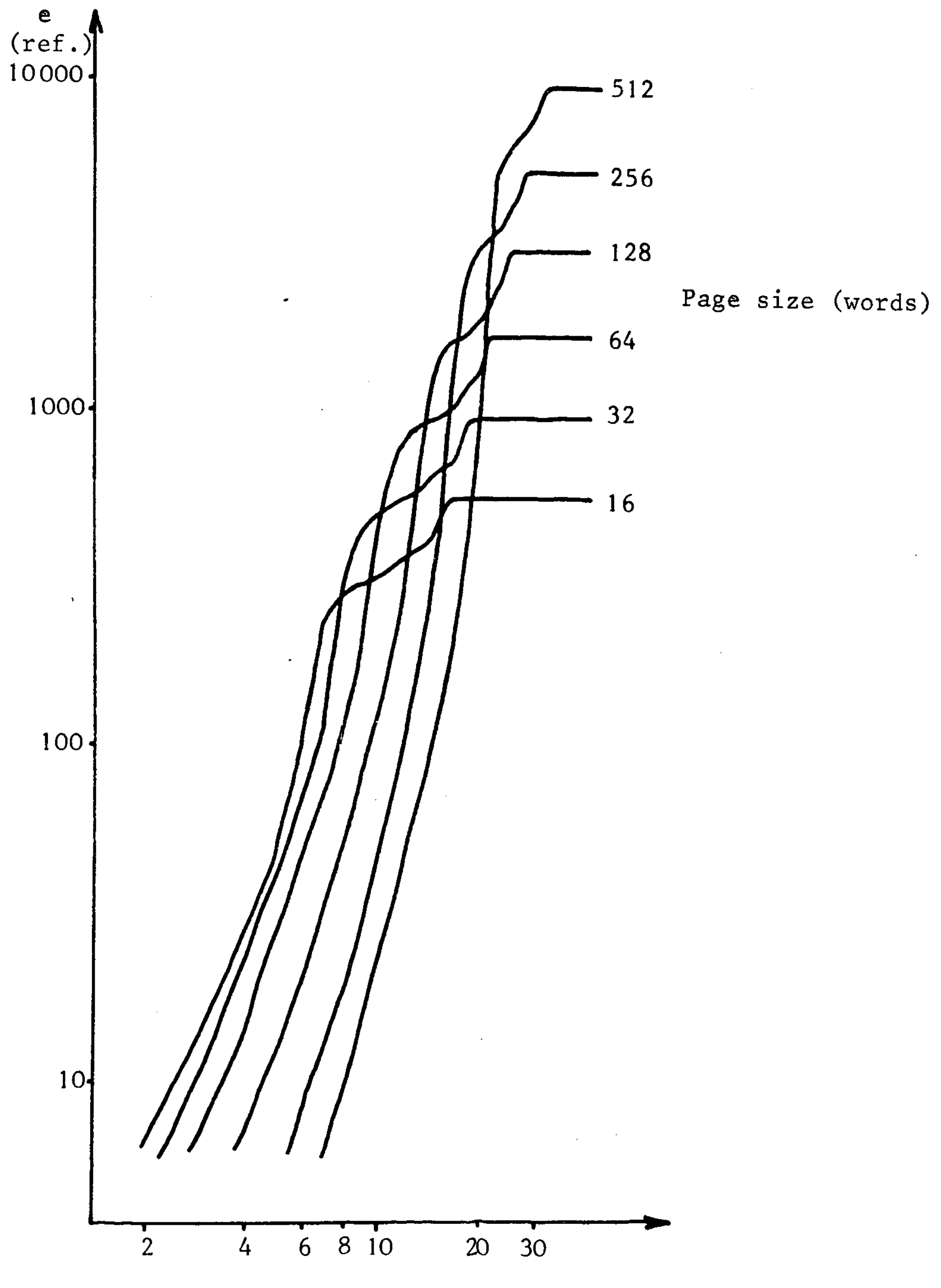

Fig.1 : Experimental LTFs for different page sizes
(FORTRAN compilation)

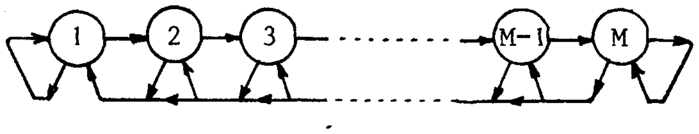

Fig. 2 : General state transition diagram

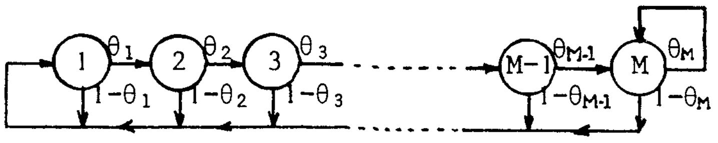

Fig. 3 : State transition diagram for model A

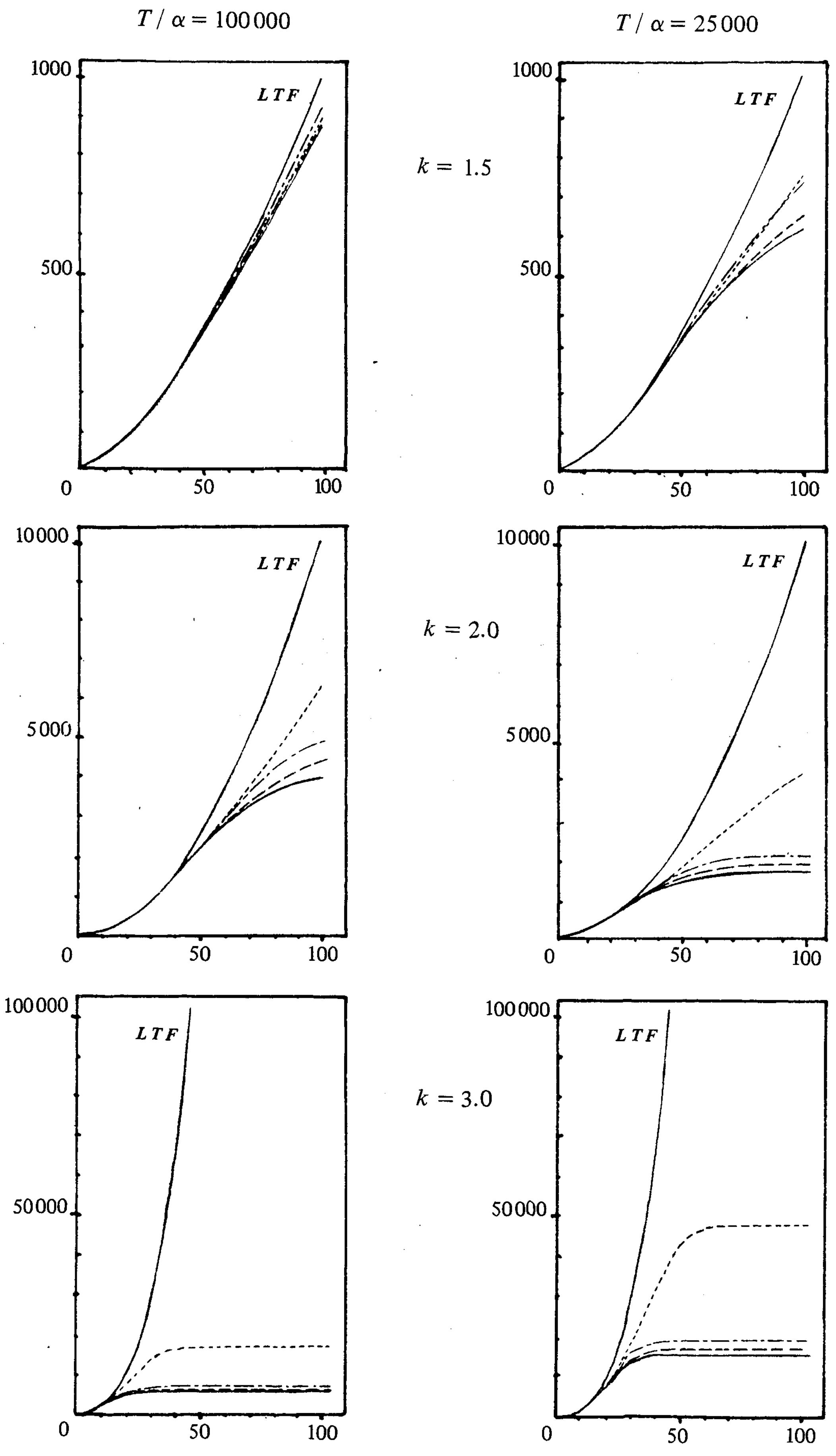

Fig. 4a : Modified life time function (e/α vs memory size)

(——mod. A ;—·—mod. B ;----mod. C(β=0.2) ;—–—mod. C(β=0.8))

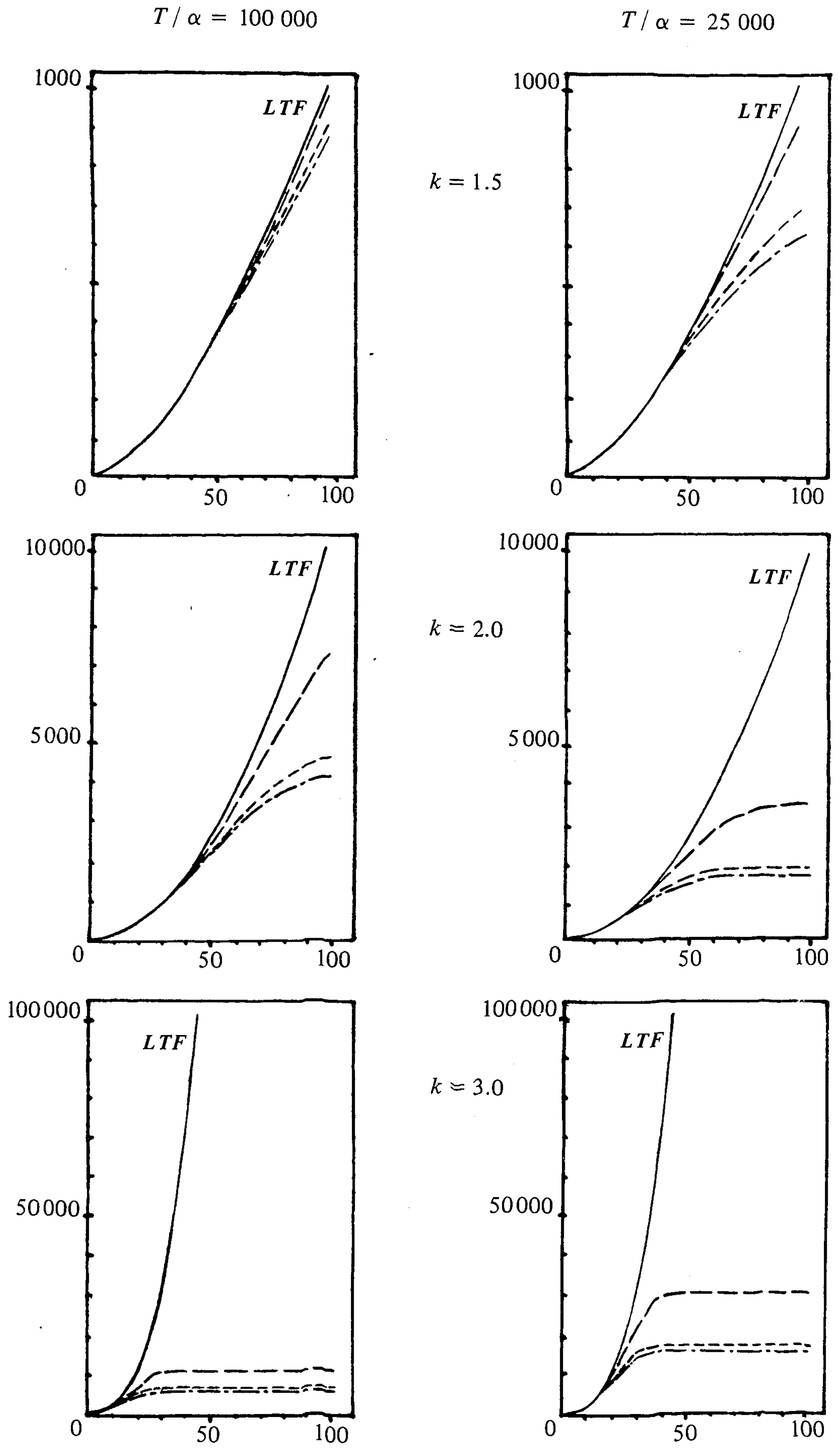

Fig. 4b : Modified life time function (e/α vs memory size)

(mod. D :——γ≈10.0 ;----γ≈1.0 ;—·—γ<0.1)

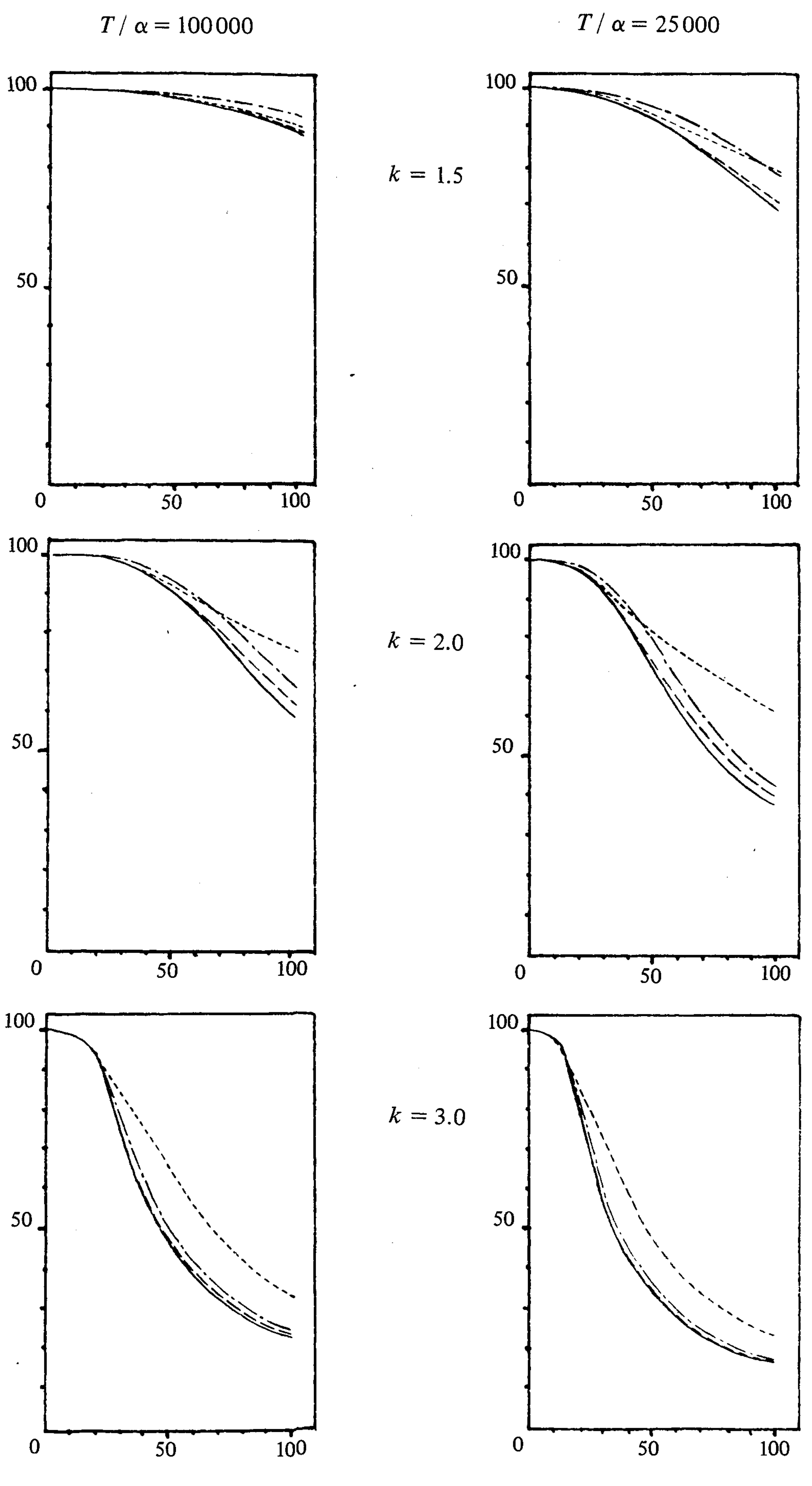

Fig. 5 : Memory utilization ratio vs number of pages allocated
(—— mod. A ;—·-mod. B ;----mod. C(β=0.2) ;—-—mod. C(β=0.8))

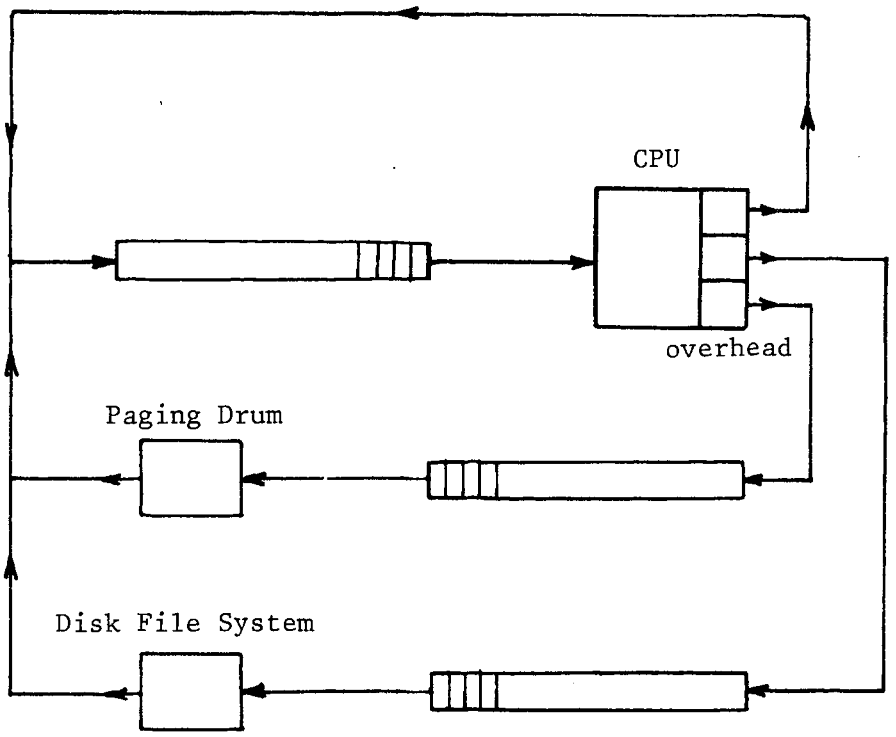

Fig. 6 : Three servers model

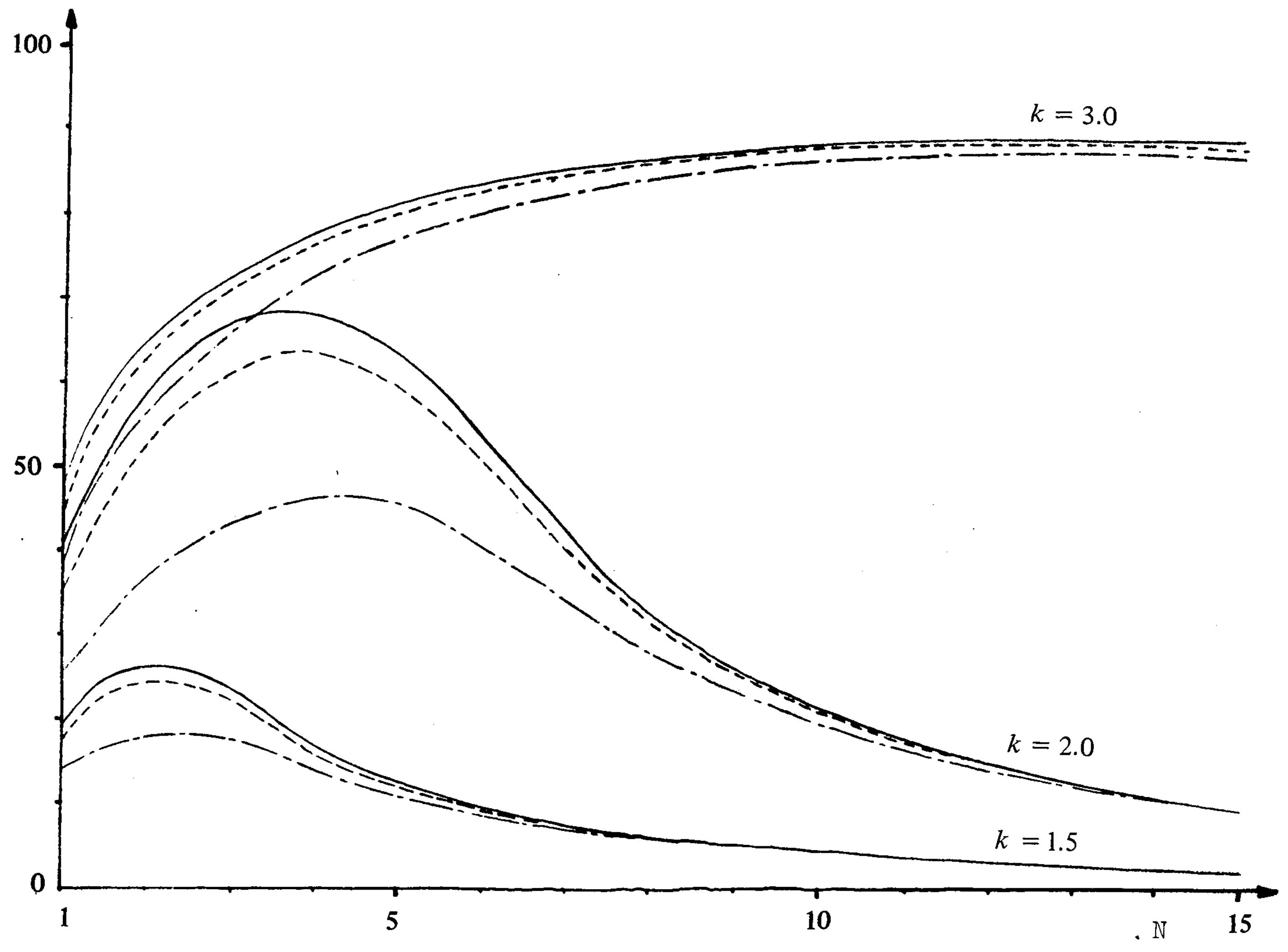

Fig. 7 : CPU utilization ratio vs multiprogramming level N

——	T=5000 msec
-----	T=1000 msec
—·—	T= 250 msec

paging drum mean service time	=	5 msec.
filing disk mean service time	=	30 msec.
mean time between I/O	=	30 msec.
number of mean memory page frames	M =	250
$e = a(M/N)^k$ where	a =	0.01 msec.
end of execution overhead	=	.5 msec.
paging overhead	=	1. msec.
I/O overhead	=	2. msec.