

7080896

Computer Architectures and Networks

Edited by
E. GELENBE
and
R. MAHL

NORTH-HOLLAND/AMERICAN ELSEVIER

TP303
G1

7660696

COMPUTER ARCHITECTURES AND NETWORKS

Modelling and Evaluation

Proceedings of an International Workshop organized by

IRIA Institut de Recherche d' Informatique et d' Automatique,
78150 Rocquencourt, France

August 12-14, 1974

edited by

E. GELENBE

Iria-Laboria

and

R. MAHL

*Ecole Nationale Supérieure
des Mines de Saint Etienne*



E7660696



1974



NORTH-HOLLAND PUBLISHING COMPANY – AMSTERDAM · OXFORD
AMERICAN ELSEVIER PUBLISHING COMPANY, INC. – NEW YORK

© North-Holland Publishing Company – 1974

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior permission of the copyright owner.

Library of Congress Catalog Card Number: 74-83728

North-Holland ISBN: 0 7204 2812 2

American Elsevier ISBN: 0 444 10711 8

PUBLISHERS:

**NORTH-HOLLAND PUBLISHING COMPANY – AMSTERDAM
NORTH-HOLLAND PUBLISHING COMPANY, LTD. – OXFORD**

**SOLE DISTRIBUTORS FOR THE U.S.A. AND CANADA:
AMERICAN ELSEVIER PUBLISHING COMPANY, INC.
52 VANDERBILT AVENUE, NEW YORK, N.Y. 10017**



PRINTED IN THE NETHERLANDS

COMPUTER ARCHITECTURES AND NETWORKS

PREFACE

The increasing complexity of computer systems and their interconnections, and the great variety of system components which can be utilized in the design of computer architectures make it increasingly difficult to identify and understand the broad issues of performance which are involved. Modelling methods, associated with measurements on existing systems and of their work-load, constitute one of the major tools for tackling this problem.

The subject of modelling and performance evaluation of computer systems is at present rapidly evolving and in the last few years significant advances and major contributions have been made though roughly fifteen years have elapsed since the first results in this area were obtained. Contacts with operations research, statistics and control theory have been enriching while certain results obtained by researchers working on computer system models have made contributions and provided new insights in queueing theory. In view of the vigor of this area at the present time, it appeared useful to us to bring together researchers in this field as well as some of the potential users of the results from industrial or academic circles. Papers were invited representing some of the main trends of research and application of modelling and measurement of computer systems. Though none of the papers in this volume have been formally refereed, many of them do represent real contributions to the area. We hope that the results presented here will appear, perhaps in more polished form, in scientific journals.

The subjects covered by the papers in this volume include modelling methodology, deterministic scheduling problems arising in multi-processing and real-time systems, probabilistic models and performance measurements of novel or classical computer architectures or operating systems, issues related to the design and performance of computer networks, and applications to system design. It is hoped that the reader will be able to come away with an impression of the state-of-the-art of the area and of the potential applications, limitations and topics for further research.

We take this opportunity to thank the people who have contributed to this workshop by presenting papers or by their presence. We also acknowledge the

support of IRIA and of Mr. A. DANZIN and Mr. M. MONPETIT. The sponsorship of the IEEE Computer Society's Technical Committee on Computer Architecture, as well as of the French Chapter of ACM is very much appreciated. We are grateful to Mlle. Thérèse BRICHETEAU and to her staff at the Public Relations Department of IRIA for their excellent support and cooperation, as well as to Mr. A. BRANDWAJN and Mr. D. POTIER, and especially to Mr. J. LEROUDIER for their help with the organization of this meeting. The cooperation of Mr. FREDRIKSSON in the preparation of the printed proceedings has been very valuable.

Erol GELENBE
IRIA-LABORIA
Rocquencourt

Robert MAHL
Ecole Nationale
Supérieure des Mines
de St. Etienne

LIST OF CONTENTS

Editor's preface E. GELENBE	VII
– Adaptive optimization of the performance of a virtual memory computer M. BADEL, E. GELENBE, J. LENFANT, J. LEROUQUIER, D. POTIER	1
– Optimal scheduling on two processors of different speeds J. L. BAER	27
– Modelling and simulation of TSS under GECOS III M. F. BAUER, K. B. IRANI	47
– Stochastic optimization methods and their value for experimenting with computing systems BRIGITTE BAUMEISTER, H. BEILNER	73
– Models and evaluation of pipeline systems D. P. BOVET, M. VANNESCHI	99
– Effects of compaction on memory utilization in multiprogramming systems J. P. BUZEN, D. B. RUBIN	113
– Optimization of segment packing in virtual memory P. P. CHEN, A. GALLO	125
– The stability problem of broadcast packet switching computer networks G. FAYOLLE, E. GELENBE, J. LABETOUILLE, D. BASTIN	135
– An analytic study of memory allocation in multiprocessing systems D. FERRARI	141
– Analytical techniques for computer networks analysis and design L. FRATTA, U. MONTANARI	155
– On approximate computer system models E. GELENBE	187
– Implementation considerations for masking rotational latency by dynamic disk allocation D. E. GOLD, D. J. KUCK	207
– Preliminary performance measurements of the CHIOS operating system P. GRANGEAN, J. LANGER, W. LYNCH, M. SCHWARTZ	239
– A fast approach to computer system performance prediction S. R. KIMBLETON	251

– Bounds for the waiting time in queueing systems H. KOBAYASHI	263
– The ALOHA broadcast packet communications system F. F. KUO	275
– Some theorems on real time scheduling J. LABETOULLE	285
– The delay network model of program behaviour J. LENFANT	299
– Performance analysis of heterogeneous multiprocessor computing systems JANE W. S. LIU, C. L. LIU	331
– A homogeneous network for data sharing E. MANNING	345
– A descriptive model for the design and implementation of computer systems J. MAZUR	355
– Networks of unreliable computers I. MITRANI	359
– Markov models of parallelism in loading and execution of a single process T. A. RYAN Jr., B. RYAN, E. G. COFFMAN, Jr.	375
– The structure of human information and some consequences for system design and assessment G. G. SCARROTT	389
– Evaluating alternatives in information system structure by simulation J. B. WYATT	405
– An Analytic model of CP-67 and VM/370 Y. BARD	419
– Modeling aspects of computer architecture B. COCHI	461
– Hierarchical techniques for development of realistic models of complex computer systems J. C. BROWNE, K. M. CHANDY, R. M. BROWN, T. W. KELLER, D. TOWSLEY	483
– New directions for network simulators N. R. NIELSEN	501
– Demand prepagating algorithms basing on a model of locality of programs O. SPANIOLO	515

ADAPTIVE OPTIMIZATION OF THE PERFORMANCE OF A VIRTUAL MEMORY COMPUTER

M. BADEL, E. GELENBE, J. LENFANT, J. LEROUQUIER, D. POTIER

IRIA

Institut de Recherche d'Informatique et d'Automatique, France

I - PREFACE

It is known that the regulation of the degree of multiprogramming as a function of instantaneous system state and program behaviour parameters (e.g. working sets) is perhaps the most important factor determining the overall performance of a virtual memory computer. The literature on this subject is abundant [5], [6], [10].

Among the regulation methods which have been proposed those which have attracted the greatest attention and which seem to have had the greatest success in practice have been derived from the working set principle [4]. Some notable exceptions to this exist and it is worthwhile to mention the virtual memory scheduler of the time-sharing system MTS running on the IBM 360/67 computer [9].

The working set strategy, as well as some others, is based on a conservative approach : the scheduler will not admit a process to share real memory if it "believes" (by an estimation of the process working set size, or otherwise) that this will result in a level of paging activity causing a serious degradation of system performance.

In this paper we take an approach which is philosophically somewhat different. We propose the design of a scheduler (henceforth called the controller) which will regulate the system in order to optimize a given performance measure, rather than simply prevent undesirable system behaviour. This follows closely the approach in control engineering practice where "regulators" have been replaced by optimizing

feedback control systems.

II - INTRODUCTION.

Though the approach presented here can be extended to systems for which the critical resource is not real memory, here we shall only be concerned with systems in which ineffective regulation leads to a phenomenon known as thrashing [6], or the reduction of useful throughput due to excessive paging activity. In order to motivate the method we propose, we shall examine in detail the dynamics of the system we wish to regulate. In Figure 1 a time sharing multiprogrammed virtual memory system is shown together with the controller. The latter consists of an estimator (EST) of system behaviour using measurements obtained at A on the various traffic patterns in the system, an optimizer (OPT) which computes decisions using the estimates provided by EST as well as with knowledge of the system state ; OPT furnishes commands to the switch (K) controlling the access into the resource loop (RL) of users waiting in the queue (Q).

The time sharing system consists of a set of terminals from which a set of users (whose number is equal to that of the terminals) generate commands. When a command is generated, the user at the terminal will remain inactive until the system provides the proper response. Symbollically, a user having generated a command enters Q and remains there until it is admitted into RL by K. The terms "user" and "user process" or just "process" will be used interchangeably. The solid lines in Figure 1 indicate the cycle a user will follow and it is in the RL that a certain number (the degree of multiprogramming) of users execute sharing a pool of resources. The dotted lines in Figure 1 represent the flow of information to the various units of the controller. The information is used by OPT to control the access of users to the resource loop so as to maximize a given performance criterion ; for instance this criterion could be the expected response time of the incoming users. In the first part of the paper we shall discuss a possible structure of the RL and present a mathematical model giving its dynamics and showing the dependence of the throughput of RL on the degree of multiprogramming. This will provide a theoretical characterization of thrashing and greater insight of the "plant" we wish to control. The mathematical results will be supplemented with data from simulation experiments using a more detailed and less idealized model.

III - MATHEMATICAL AND SIMULATION MODEL OF THE RESOURCE LOOP.

3.1 - The Model.[12]

Consider the model described in Figure 2. It represents the system architecture of the multiprogrammed paged virtual memory computer under consideration. The system consists of a CPU, a secondary memory device (SM) (e.g. fixed head disk or drum),

and a file disk (FD). A queue of requests is associated with each device; the order in which these requests are satisfied is not taken into consideration.

N statistically identical and independent processes (users) execute in the system, so that if at any time n_0, n_1, n_2 are the numbers of processes in the CPU, SM, FD queue respectively, then

$$(1) \quad N = n_0 + n_1 + n_2$$

In the model, the behaviour of processes is characterized by a compute time followed by either a page fault (after which the process enters the SM queue) or an input-output (file request) in which case the process enters the FD queue. Processes which terminate their service at the SM or at the FD return to the CPU queue. The feedback loop around the CPU and its queue represents processes which depart from the system but which are immediately replaced by a new process, maintaining a constant degree of multiprogramming.

Let τ be the random variable representing uninterrupted computation at the CPU.

Let t_1, t_2 be the random variables representing the compute time between two successive page faults, or I/O operations, respectively. Suppose that t_1, t_2 have probability distribution functions $F_1(t), F_2(t)$. Let also t be the random variable representing total compute time of a process, with distribution function $F_3(t)$. We shall assume, in order to make use of the "memoryless" property, that

$$(2) \quad \begin{aligned} F_1(t) &= 1 - e^{-\frac{t}{q}} \\ F_2(t) &= 1 - e^{-\frac{t}{r}} \\ F_3(t) &= 1 - e^{-\frac{t}{c}} \end{aligned}$$

the assumption being that t_1, t_2, t_3 are independent. Because of the memoryless property of the exponential distribution, the random variable τ is distributed as

$$\min \{t_1, t_2, t_3\}$$

and we then obtain

$$(3) \quad \begin{aligned} F(t) &= \text{Prob} [\tau \leq t] \\ &= 1 - \text{Prob}[t_1 > t \text{ and } t_2 > t \text{ and } t_3 > t] \\ &= 1 - [1 - F_1(t)][1 - F_2(t)][1 - F_3(t)] \\ &= 1 - e^{-t \left[\frac{1}{q} + \frac{1}{r} + \frac{1}{c} \right]} \end{aligned}$$

Therefore τ is exponentially distributed with expected value

$$(4) \quad E\{\tau\} = \frac{1}{u_0} = \frac{1}{\frac{1}{q} + \frac{1}{r} + \frac{1}{c}}$$

Note that τ is the service time at the "server" representing the CPU in the queueing network model of Figure 1.

To obtain p_1 and p_2 , the probabilities that a program leaving the CPU will direct itself to the SM device or the FD, respectively, we proceed as follows :

$$(5) \quad p_1 = \int_0^{\infty} \text{Prob} [t-dt < t_1 \leq t \text{ and } t_2 > t \text{ and } t_3 > t]$$

where the integral is over all values of time t , since it is the probability that τ and t_1 coincide. We then have

$$(6) \quad p_1 = \int_0^{\infty} dt \frac{1}{q} e^{-\frac{t}{q}} e^{-\frac{t}{r}} e^{-\frac{t}{c}} \\ = \frac{1}{u_0 q}$$

Similarly, we shall have

$$(7) \quad p_2 = \int_0^{\infty} dt \frac{1}{r} e^{-\frac{t}{r}} e^{-\frac{t}{q}} e^{-\frac{t}{c}} \\ = \frac{1}{u_0 r}$$

and

$$(8) \quad p_0 = 1 - p_1 - p_2 = \frac{1}{u_0 c}$$

We assume that the service times at the SM and the FD are exponentially distributed with expected values $\frac{1}{u_1}$, $\frac{1}{u_2}$, respectively.

3.2 - THE EQUATIONS DESCRIBING SYSTEM BEHAVIOUR.

We would like to use the preceding assumptions regarding the service time distributions, and their relationship to program behaviour, in order to compute the throughput of the system as a measure of system performance. The throughput will be defined as the number of programs processed by the system per unit time, in the long run (at steady state).

The behaviour of the system at steady state is completely characterized by the joint probability distribution $p(n_0, n_1, n_2)$ of the number of processes in the three queues

It can be shown that $p(n_0, n_1, n_2)$ exists and is independent

of initial conditions. Moreover, the model is a particular case of Jackson's networks [8] so that we have the stationary solution

$$(9) \quad p(n_0, n_1, n_2) = \frac{1}{G(N)} \left(p_1 \frac{u_0}{u_1} \right)^{n_1} \left(p_2 \frac{u_0}{u_2} \right)^{n_2},$$

where

$$(10) \quad G(N) = \sum_{n_1, n_2} \left(p_1 \frac{u_0}{u_1} \right)^{n_1} \left(p_2 \frac{u_0}{u_2} \right)^{n_2}$$

System throughput is obtained as follows. Each process makes, on the average, a total of $\frac{1}{p_0}$ passes through the CPU, requiring thus a total average compute time $\frac{1}{p_0 u_0}$. In a total time T , the CPU is busy for a time $A_0 T$, where A_0 is the probability of a non-empty CPU queue. A_0 will be called the CPU utilisation and is given by

$$A_0 = \sum_{\substack{1 \leq n_0 \leq N \\ 0 \leq n_1 < N}} p(n_0, n_1, n_2)$$

and is obtained directly from (15) (see [14]) as

$$(11) \quad A_0 = \frac{G(N-1)}{G(N)}$$

Therefore in a time T , the average number of programs processed is $A_0 T u_0 p_0$ and the throughput is $\theta = A_0 u_0 p_0$. From (8) we have

$$(12) \quad \theta = \frac{1}{c} A_0$$

Let A_1, A_2 be the utilization factors of the SM device and FD, respectively. It can be easily shown [14] that

$$A_1 = p_1 \frac{u_0}{u_1} A_0$$

$$A_2 = p_2 \frac{u_0}{u_2} A_0$$

3.3 - THE EFFECT OF MEMORY SHARING.

We propose to use our model to analyze the effect of memory sharing in the system. We therefore need a model relating $F_1(t)$ to the amount of space allocated to each program.

Experimental evidence [1] indicates that the mean time between page faults q for a process executing in memory space m is approximated by a function of the form

$$(13) \quad q = a m^k$$

where a depends on the processing speed as well as on program characteristics while k depends on program locality and on the memory management strategy (e.g. the page

replacement algorithm). Belady and Kuehner [1] indicate that k is in the range $1.5 < k < 2.5$, while Saltzer [11] claims that it is close to unity, r is assumed independent of m or N , which seems reasonable.

We shall assume that total primary memory available is of size M and that it is equally shared among processes :

$$(14) \quad m = \frac{M}{N}$$

Therefore the system considered will be used to model either a batch system with a paged virtual memory or the behaviour of the programs sharing PM in a time-shared system. In the latter case it becomes clear that the interest of a stationary analysis (as carried out in most of the literature) is limited to the case when the stationary state is reached rapidly.

Numerical results obtained from our model are reported in Figures 3 and 4 in which the changes obtained in A_0 , the CPU utilization, versus N (the degree of multiprogramming) are shown for a wide range of system parameters. The mean compute time c is kept constant throughout the examples at 1 sec., and α is set to $\alpha=0.04$. In figure 3 the effect of the degree of multiprogramming N on CPU utilization A_0 (and hence on throughput $\theta = \frac{1}{c}$) is shown for a set of system parameters with a primary memory size $M=128$ pages. The effect of varying locality of the programs (the parameter k) is examined in Figure 4 with the remaining parameters kept constant. We see that an improvement in program behaviour (increasing k from 1.5 to 2.5) can produce a dramatic increase in the optimal degree of multiprogramming as well as an improvement of the throughput.

As a whole, a comparison with results of measurements on real systems of similar effects [10] indicate that although our curves have a similar form, they overestimate the CPU utilization (because interrupts and system overhead are not taken into account). We also notice that the "tails" of our curves decrease less rapidly than measured curves ; we are unable to explain the reason for this phenomenon. Notice that if the distribution of t_j is Erlang with parameters (c,k) , then the model is still valid since we can suppose that the program passes k times (rather than just once) through the system. The average compute time is then kc and the throughput is

$$\theta = \frac{1}{kc} A_0$$

and (9) and (10) remain valid.

3.4 - SIMULATION RESULTS.

Different simulation runs have been performed for various modifications of the basic model, the aim being to observe the influence of non-exponential service parameters as well as system overhead. The simulation model has been written in SIMULA 67 with an observed run time to model time ratio of roughly six. The simu-

lation model has been "validated" by running it with exponential service times and no overhead so as to be able to compare its behaviour, at least under these conditions, with known analytical results.

In our simulation the total compute time of each process is still exponentially distributed. A constant CPU time, representing system overhead, has been added to each CPU service termination. The duration of the overhead is dependent on the reason for the termination of the CPU pass : it is

- a_0 if the program has terminated its computation and has been replaced by a new program in the CPU queue,
- a_1 if a page fault has occurred,
- a_2 if an I/O operation has been requested.

The simulation results have been plotted on Figures 3(a) and 3(b), where the light overhead conditions correspond to $a_0 = 1$ ms, $a_1 = 0.5$ ms, $a_2 = 2.0$ ms. The heavy overhead conditions correspond to $a_0 = 1$ ms, $a_1 = 2$ ms, $a_2 = 2$ ms. Since p_0 is very small in the examples considered, the choice of a_0 has little influence on the results which are affected most by a_1 and a_2 . The CPU utilization shown in Figures 3(a) and 3(b) is the fraction expended for user programs ; the computation expended in overhead having been deduced. On Figure 3(c) we show the real CPU utilization (including the overhead). In the examples we have examined, we have noticed that the overhead and the non-exponential service times at the drum and disk do not change significantly the region in which useful CPU utilization is maximised. The significant influence is that of the overhead and not of the non-exponential service times.

IV - THE ADAPTIVE CONTROL SYSTEM.

4.1 - The basic design considerations.

The problem for the adaptive controller is to maintain the system in operation so that at all times the measure of performance is optimized. A definition of this measure is given in the next section. The basic physical assumptions concerning the system to be controlled are the following :

A1 - The measure to be maximized (the criterion) is a function only of the degree of multiprogramming n , or equivalently n is the only parameter the controller is allowed to modify.

This assumption of course simplifies the problem considerably. Other parameters which could be used to improve the system performance could be related to various scheduling disciplines making a choice about the user jobs to be admitted in the RL. The scope of this study has been limited purposely to the control of the degree of multiprogramming.

A2 - The measure has a single extremum as a function of n . This will be justified in the next section.

A3 - The characteristics of user processes vary "slowly" with respect to the arrival rate of processes to the system and to the internal time constants (e.g. service times of resources) of the RL.

As a consequence, the variations of the characteristics of user processes will modify the "shape" of the criterion as a function of n . OPT is built so as to adapt itself to those modifications.

Some broad considerations govern the design of the controller.

D1 - The degree of multiprogramming n is regulated to the value n_0 at which no more increase is sensed in the criterion while increasing n .

D1 is derived from the fact that the criterion has a single maximum (A2). However, in some cases the criterion maximum value is reached for many consecutive values of n , and those values of n are not equivalent when the response time is concerned. As a matter of fact, due to overhead effects through paging and quantum allocation, the users' response time increases with n .

D2 - The estimator (EST) sensing the "shape" of the criterion as a function of n weighs more heavily recent events. On the other hand the optimizer (OPT) includes a certain inertia in its reactions to information furnished by EST so as to eliminate the effects of "noise".

D3 - All actions of the controller are a consequence of normal system events. Periodic measurements or control actions are excluded.

The purpose of D3 is to limit excessive overhead which would occur if measurements or control actions took place at frequent periodic intervals. It also insures that no important events are overlooked.

The control scheme considered in this paper takes only into account the introduction of processes into RL and not the elimination of processes (for instance, of those whose paging activity is excessive). Processes are assumed to leave RL due to events which are independent of the control.

4.2 - The criterion.

The main objectives we want to achieve through the dynamic optimization of the degree of multiprogramming are to maximise the utilization factor of the different processors and to minimize the users response time.

Let us consider a RL with K resources. Users enter the RL and are processed using the different resources of the loop. Given a time interval $[t_{i-1}, t_i]$ during which only one process uses the resources, the sum of the different services ts_k received by the user on each resource K is equal to $t_i - t_{i-1}$ (ts_k does not include waiting

times). When waiting times and overhead are inexistent, then

$$\frac{\sum_{k=1}^K \tau s_k}{t_i - t_{i-1}} = 1$$

If more than one process shares the resources, the sum of all the services τs_k received by the users can exceed $t_i - t_{i-1}$ due to the parallelism of the resources. We now define

$$\tau_k^i = \frac{\tau s_k}{t_i - t_{i-1}}$$

which represents the utilization factor of resource k by the users during the time interval $[t_{i-1}, t_i]$. From τ_k^i we define

$$x_i = \sum_{k=1}^K \tau_k^i c_k$$

where c_k is a constant representing the relative importance of the k -th resource. x_i will be used as our criterion for the dynamic optimization of the degree of multiprogramming. We may note that x_i measures the effectiveness of the use of the parallelism of the set of resources, and expresses that more than one unit of service time can be processed in one unit of real time. In the following, x_i will be called "dilatation".

From the analytical model of the resource loop presented in Section 3 it can be seen that the utilisation factors of the different processors are proportionnal to each other and that there exists a value n_0 for which the utilization factors are maximum. Extensive simulation runs showed that this is still true when exponential assumptions are dropped and overheads are taken into account (see Fig. 3a,b). The main effect of overheads is to flatten the upper part of the curve as shown on Fig. 3a. It should be pointed out that x_i is computed using only services required by the users: CPU time and I/O processing time, excluding overheads and SM service. This has been already partly explained in section 4.1 when we justified design consideration D1.

4.3 - Design of controller functions.

In this section we present the essential functions realized by the controller. They have been implemented in the simulation model of a virtual memory time-sharing system described in Section 5 where detailed simulation results are also given. A theoretical justification for each of the design decisions cannot be given because of the complexity of the problem, but (as in most engineering design) these are supported by evidence that the approach is reasonable and that the resulting performance is good.