Operating Systems J. Stockton Gaines Editor

Decomposability, Instabilities, and Saturation in Multiprogramming Systems

P.J. Courtois
MBLE Research Laboratory, Brussels

A step-by-step approach to model the dynamic behavior and evaluate the performance of computing systems is proposed. It is based on a technique of variable aggregation and the concept of nearly decomposable systems, both borrowed from Econometrics. This approach is taken in order to identify in multiprogramming paging systems (i) unstable regimes of operations and (ii) critical computing loads which bring the system into states of saturation. This analysis leads to a more complete definition of the circumstances in which "thrashing" can set in.

Key Words and Phrases: multiprogramming, paging, performance evaluation, saturation, instabilities, thrashing, aggregation, system levels, hierarchy, networks of queues

CR Categories: 4.31, 4.32, 8.1, 8.2

Author's address: Avenue Em. Van Becelaere 2, B-1170 Brussels, Belgium.

1. Introduction

Various models already exist which enable the performance of different components of computer and operating systems to be predicted: models of allocation mechanisms for processors and storage devices, of page replacement strategies, of interference between input-output and central processing unit operations, etc.

However, except perhaps for the simulation approaches recommended in [13] and [17], no framework yet exists in which all such fragments of theory could be embodied so that one could predict with sufficient and known precision how the whole of a system of a specified design will actually react to a certain environment of users. This means that the designer has no other alternative than to build his system and see how it behaves. This approach results in a huge amount of expensive experimentation which could be reduced if global models of computer system performance existed.

The first part of this paper introduces rather informally the more important aspects of an attempt to set up such models; a more rigorous and more complete study is developed in [4]. This attempt is based on the concept of near-decomposability and on a technique of variable aggregation, both borrowed from Econometrics. We show how this technique can advantageously be used to analyze the dynamics of stochastic systems with shared resources, these systems being of arbitrarily great size and complexity. In the second part an aggregative model is used to identify in multiprogramming paging systems: (i) regimes of operations which are unstable; and (ii) critical computing loads which bring the system into states of saturation.

This second part should also serve as an illustration to the first one by showing that aggregation is not only efficient in obtaining numerical results when a large number of parameters are involved but also helpful in gaining insight and conceptual clarity on the parts played by these parameters.

2. Variable Aggregation and Near-Decomposability

Variable aggregation is a technique which has been most explicitly used by economists to analyze large and complex systems, viz. systems which, roughly speaking, are represented by a large number of state variables interacting on each other in many various ways. This technique is based on the recognition that in most of these systems, state variables can be classified into a small number of groups such that

—interactions *within* groups can be studied as if interactions *among* groups did not exist.

—interactions *among* groups can be analyzed without referring to the interactions *within* groups.

This working hypothesis is rigorously, but at the same time, trivially correct if variables within a group depend only on variables of the same group. The system is then said to be *completely decomposable* into as many subsystems as there are groups. In 1961, H.A. Simon and A. Ando [14] explored circumstances under which this technique still yields good approximations when interactions among groups do exist but are weak compared to the interactions within groups. Such systems were qualified as being *nearly completely decomposable.*

Simon and Ando showed that in such systems *short-run dynamics* can be distinguished from *long-run dynamics*; they proved that two properties follow from this distinction [14]:

1. In the short-run dynamics, a local equilibrium is reached by the strong interactions within each subsystem almost independently of the other subsystems.
2. In the long-run dynamics, the weak interactions among groups make themselves felt and the whole system moves towards a global equilibrium maintaining approximately the relative equilibrium values attained by the state variables of each subsystem at the end of the short-run dynamics period.

These properties give an indication as to which variables should be aggregated in nearly completely decomposable systems: the short-run dynamics may approximately be analyzed separately in each subsystem, and the local equilibrium of each subsystem may be represented by an (a few) aggregative variable(s); the long-run dynamics of the whole system may then be analyzed as a set of interactions among these variables, the interactions within each subsystem being ignored. In the following, these subsystems will be referred to as aggregates.

3. A Hierarchy of Aggregate Resources

3.1 Application

Let us apply these concepts to a simple model of a network of queues which was first analyzed by J.R. Jackson [9]. Consider a set of $(L + 1)$ resources denoted $R_0, R_1, \ldots, R_L$. The time to complete a service at resource R_l is assumed to be a random variable exponentially distributed with mean equal to $1/\mu_l$, $l = 0, \ldots, L$.

A customer (job) in this system is supposed to follow a random walk: a customer who completes service at resource R_m immediately requests service from resource R_l with transfer probability p_{ml} $(\sum_{l=0}^{L} p_{ml} = 1)$.

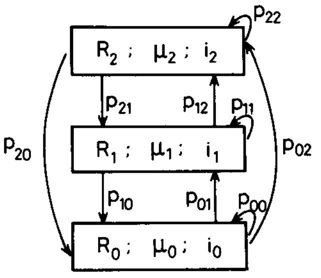
Let N be the number (finite and constant) of customers in the system and $i_l(t)$ the number of customers in service or in queue at resource R_l at time t $(\sum_{l=0}^{L} i_l(t) = N)$.

The state of the system at time t is characterized by the vector $[i_0(t), \ldots, i_l(t), \ldots, i_L(t)]$. The number of distinguishable states is equal to the number of partitions of N customers among $(L + 1)$ sets, viz. $\binom{L+N}{L}$. Since the service times are exponentially distributed, the probability of more than one service being completed during an elementary time interval is negligible. Thus the only possible transitions during an elementary time interval $(t, t + 1)$ are, for some pair (l, m), from a state $[i_0(t), \ldots, i_l(t), \ldots, i_m(t), \ldots, i_L(t)]$ to state $[i_0(t), \ldots, i_l(t) - 1, \ldots, i_m(t) + 1, \ldots, i_L(t)]$. The probability of such a transition is equal to $\mu_l p_{lm}$ if $i_l(t) \geq 1$, and zero otherwise.

We assume that the process so defined is irreducible, viz. that the system can go from any given state to any other state within a finite number of transitions. Since the model corresponds to an irreducible, finite state Markov process, there exist equilibrium probabilities, say $s_l(i_l)$, $i_l = 0, \ldots, N$; $l = 0, \ldots, L$ of i_l customers being in queue or in service at resource R_l, which are independent of time and of the system initial conditions. Our purpose is to evaluate this steady-state distribution.

The system may be represented by an oriented graph; an example for three resources is given in Figure 1.

Fig. 1.



Suppose now that the order of magnitude of the transition probabilities $\mu_0 p_{01}$ and $\mu_1 p_{10}$ between R_0 and R_1 is greater than the order of magnitude of the transition probabilities between R_2 on the one hand and R_0 or R_1 on the other hand. Such a system may be analyzed in the following way [3, 4].

A local statistical equilibrium will be reached in the distribution of customers between R_0 and R_1 before interactions with R_2 will make themselves felt. To approximate this distribution, we may therefore assume that: (i) interactions with R_2 do not exist; and (ii) the number, say n_1, of customers in queue or in service in the aggregate (R_0, R_1) remains constant (for instance equal to its initial value at time $t = 0$).

Let $\pi_1(n_0 \mid n_1)$ be the conditional probability that n_0 customers be in queue or in service at R_0 and $(n_1 - n_0)$ at R_1, on condition that there is a total number n_1 of customers in (R_0, R_1). This distribution may easily be obtained for all possible values of n_1 $(1 \leq n_1 \leq N)$ as the distribution of the congestion in an $M \mid M \mid 1 \mid n_1$ queue with finite population n_1.

After this local equilibrium is reached, the probability of a customer moving from aggregate (R_0, R_1) to resource R_2 during an elementary time interval is

$$\mu_1[1 - \pi_1(n_1 \mid n_1)] \times p_{12} + \mu_0[1 - \pi_1(0 \mid n_1)] \times p_{02} = \psi_{12}(n_1),\ n_1 = 1, \ldots, N;\ \psi_{12}(0) = 0.$$

Since this internal equilibrium in the aggregate (R_0, R_1) is maintained even when the interactions with R_2 have had time to make themselves felt, we can study these interactions in terms of the aggregative variables $\psi_{12}(n_1)$, $n_1 = 0, \ldots, N$. Here, we need only to consider an $M \mid M \mid 1 \mid N$ queueing system, with finite population N, service rate $\psi_{12}(n_1)$, $n_1 = 0, \ldots, N$ (which has the peculiarity of being congestion dependent), and input rate $\mu_2(p_{20} + p_{21})$. We will obtain the stationary distribution $\{\pi_2(n_1 \mid N)\}_{n_1=0}^{N}$, $\pi_2(n_1 \mid N)$ being the stationary probability of having n_1 customers in the aggregate and $N - n_1$ in queue or in service at R_2.

If R_0, R_1, R_2 were part of a larger system in which they could be considered as one aggregate, the interactions between this aggregate and the remainder of the system could be studied in terms of the aggregative probabilities ($l \neq 0, 1, 2$):

$$\psi_{2l}(n_2) = \mu_2 p_{2l}(1 - \pi_2(n_2 \mid n_2)) + \sum_{n_1=1}^{n_2} \pi_2(n_1 \mid n_2)\psi_{1l}(n_1),\ n_2 = 1, \ldots, N.$$

We could consider in this way as many levels of aggregation as there are resources in the system, aggregation variables being defined at each level in terms of the variables of the immediately lower level.

So far, proceeding from the inner to the outer aggregate, only conditional probabilities $\pi_l(n_{l-1} \mid n_l)$, $l = 1, \ldots, L$, have been obtained. The marginal probabilities $s_l(i_l)$ can be deduced from these probabilities by proceeding in the reverse order [i]; if $a_l(n_{l-1})$ denotes the unconditional probability of a total of n_{l-1} customers being in queue or in service at resources $R_0, R_1, \ldots, R_{l-1}$, we have:

$$a_L(n_{L-1}) = \pi_L(n_{L-1} \mid N),\ s_L(i_L) = \pi_L(N - i_L \mid N),\quad n_{L-1}, i_L = 0, \ldots, N,$$

and for each level $l = L - 1, \ldots, 1$:

$$a_l(n_{l-1}) = \sum_{n_l=n_{l-1}}^{N} a_{l+1}(n_l)\pi_l(n_{l-1} \mid n_l),$$

$$s_l(i_l) = \sum_{n_l=i_l}^{N} a_{l+1}(n_l)\pi_l(n_l - i_l \mid n_l); \tag{1}$$

at level 1, whose only constituents are resources R_1 and R_0, we have $s_0(i_0) \equiv a_1(i_0)$, $i_0 = 0, \ldots, N$.

Using Little's formula [11], the mean response time of a resource R_l, defined as being the mean time spent by a customer in queue or in service at this resource, may easily be derived from the probabilities $s_l(i_l)$.

3.2 Discussion

The advantages of this aggregation technique result essentially from the fact that the analysis of an n-state system (in the above example, $n = \binom{N+L}{L}$) is reduced to the analysis of a limited number ($L \times N$ above) of independent subsystems, each of these having a smaller number ($\leq (N + 1)$ above) of states.

As far as queueing network analysis is concerned for example, the above developments yield the marginal probabilities $s_l(i_l)$ without having to resort, as is required in [2] and [9], to the resolution of an $(L + 1) \times (L + 1)$ linear system of equations with matrix $[\mu_i p_{ij}]$. And the same approach can be followed (see [4]) to analyze more complex networks in which service rates and/or transfer probabilities are state-dependent and where service times are arbitrarily distributed. R. Muntz and F. Baskett [12] have recently analyzed networks of queues with different classes of customers. Aggregates in such networks must be defined not only in terms of resources but also of classes. The advantages of aggregation have still to be assessed in this case.

Aggregation also gives some insight into the transient behavior of the system since it makes explicit the various local equilibrium states attained by each aggregate at different stages of the evolution of the whole system. This point will be illustrated in the next section.

Another important possibility offered by this state-space partitioning is that it opens the way to the integration of different methods of analysis (queueing theory, simulation, deterministic models, ...), each aggregate being separately analyzed by the most appropriate approach.

Of course, the results obtained are only approximations. But the degree of approximation is known and predictable. It can be proved (see [5]) that the error made at each level of aggregation remains of the same order of magnitude as the ratio of the inter-aggregate interactions to the intra-aggregate interactions, and is primarily dependent upon the degree of cohesiveness, or irreducibility of the aggregates. A method to estimate the aggregate irreducibility and the degree of approximation in function of the inter- and intra-aggregate interactions is developed in [5].

The analysis in Section 3.1 shows also that a nearly decomposable system of shared resources must be a well-balanced system in which faster resources are more frequently used than slower ones. This gives aggregation techniques a wide range of applicability. In the field of computing systems, a storage hierarchy, for example, is nearly decomposable, since the slower memory levels will necessarily be the less frequently referred to if the goal of the system is to minimize the average access time of the hierarchy. The more pertinent example to be mentioned here is the THE operating system [7], built as a hierarchy of "abstract machines" piled one on top of the other. The role of an abstract machine is to "rebuild" [8] a hardware resource into a more convenient "abstract resource" which will be used by the upper levels. To do so, an abstract machine may only resort to the abstract resources created at the lower levels. This restriction is intended to facilitate the step-by-step construction and verification of the system. We could add the step-by-step evaluation as well. Indeed, to operate efficiently, an abstract machine should make use of resources at least as fast as the hardware it is abstracting from; the faster resources (fast being measured by the grain of time appropriate to describe the activity of this resource) must therefore be implemented at the lower levels; these are also the most heavily used levels since they may be called by the whole structure above. In such an organization where the faster resources are also the more frequently requested, a level of abstraction is thus likely to be assimilable to a level of aggregation.

4. Instabilities in Systems with Shared Resources

In this last section, an aggregation technique is used to show that the values attributed to some classical parameters of a multiprogramming system may render the working conditions of this system unstable.

4.1 The Model

Let us start with a simple model of a multiprogramming paging system:

1. A finite number N of active user terminals originate with Poissonian rate λ requests for program execution (jobs). There may exist at most one job per terminal at a time in the system.
2. Jobs are executed in main memory on a multiprogrammed basis. Let J be the current number of jobs being multiprogrammed in the system.
3. Pages which cannot be contained in main memory are located in an auxiliary memory level from which they are loaded on a page on demand strategy.

Therefore, at any moment in time, multiprogrammed jobs are in one of three states: *ready*, i.e. demanding but not receiving the control of the processor; *running*, i.e. receiving the control of the processor; *suspended*, i.e. waiting for a page transfer from auxiliary to main memory to be completed.

We make the following additional assumptions:

1. At any time, each multiprogrammed job is allocated an equal number *entier* (tot/J) of page frames in main memory, "tot" being the total number of page frames available in main memory. This number is also the maximum number of distinct pages a multiprogrammed job may accumulate in main memory; whatever page replacement strategy is used, the page fault rate of a *running* job is therefore a nondecreasing function of J [16], say, $\mu_0(J)$, $J = 1, \dots, N$.
2. We assume that the requests for page transfer from auxiliary to main memory are not necessarily served on a FIFO basis, but in some order which depends upon the current state of the auxiliary memory so as to optimize the overall page transfer rate. This rate becomes in this case a function of the number of pending transfer requests (see e.g. [15]), viz. of the number, say, $i_1 (i_1 = 0, \dots, J)$, of *suspended* jobs. Let $\mu_1(i_1)$ denote this page transfer rate.

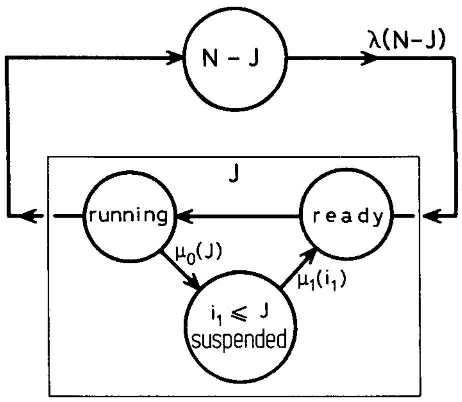
This simple model may be schematized by the state-transition diagram of Figure 2. We suppose that $\mu_0(J)$ and $\mu_1(i_1)$ are known for all possible values of J and i_1.

4.2 Decomposability of the Model

The rates of transition between *running*, *suspended*, and *ready* states will usually be much higher than the rates at which jobs are created and completed. The model is therefore nearly completely decomposable into the set of N terminals on the one hand and an aggregate corresponding to the subsystem cpu-main memory-auxiliary memory on the other hand.

The short-term statistical equilibrium reached by

Fig. 2.



the aggregate is approximately independent of the interactions with the terminals; this means that in the short run J can be considered as remaining approximately constant in the aggregate. Moreover long-term equilibrium values for J can be evaluated in terms of aggregative variables characteristic of this short-run equilibrium.

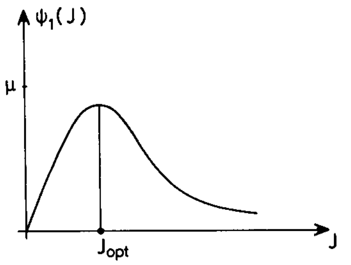
4.3 The Aggregate Short-term Equilibrium

The short-term equilibrium distribution of jobs among *ready*, *running*, and *suspended* states can thus be obtained as the steady state distribution of the congestion in an $M \mid M \mid 1 \mid J$ queueing system with fixed and finite population J, service rate $\mu_0(J)$, and congestion-dependent input rate $\mu_1(i_1)$; $i_1 = 0, \dots, J$; and this for all possible values of J: $0 < J \leq N$.

In this system, the probability, say $\pi_1(0 \mid J)$, that the server is idle is the probability that the cpu is idle, the J jobs being *suspended*. If the average total cpu time required by a job is $1/\mu$, the output rate of the aggregate in jobs per time unit, say $\psi_1(J)$, will be $\psi_1(J) = \mu(1 - \pi_1(0 \mid J))$.

It is well known that the function $\psi_1(J)$ usually takes the shape displayed in Figure 3. The optimal degree of

Fig. 3.



multiprogramming, J_{opt}, results from two counteracting effects which take place when J increases: the decrease of $\pi_1(0 \mid J)$ on the one hand, and the increase of the page fault rate as the space available to each job in main memory shrinks.

The deterioration of the aggregate output rate $\psi_1(J)$ which results from the fractioning of main memory between too many jobs may be kept under ac-

Fig. 4.

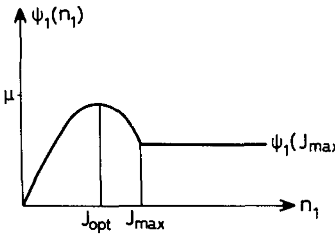


Fig. 5.

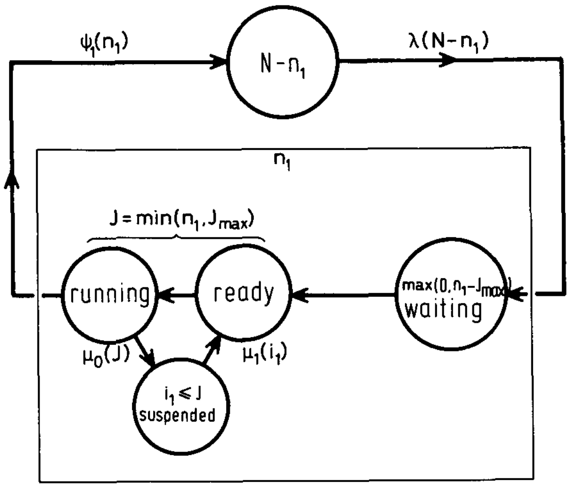
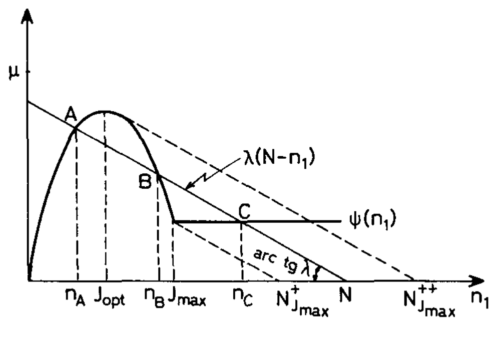


Fig. 6.



ceptable limits by imposing an upper bound, say $J_{\max}$, upon the number of jobs which may simultaneously be multiprogrammed.

In this more refined model call n_1, $0 \leq n_1 \leq N$, the total number of jobs in the aggregate. Whenever $n_1 > J_{\max}$, $(n_1 - J_{\max})$ jobs are kept *waiting* (an additional state for a job) outside the multiprogram mix. So that we have

$$\begin{aligned}\mu_0(n_1) &= \mu_0(J), \text{ for } n_1 = J \leq J_{\max},\\ &= \mu_0(J_{\max}), \text{ for } n_1 > J_{\max}.\end{aligned}$$

With μ and $\mu_1(i_1)$, $i_1 < J \leq J_{\max}$, keeping the same values, we have also:

$$\begin{aligned}\pi_1(0 \mid n_1) &= \pi_1(0 \mid J), n_1 = J \leq J_{\max},\\ &= \pi_1(0 \mid J_{\max}), n_1 > J_{\max},\end{aligned}$$

and

$$\begin{aligned}\psi_1(n_1) &= \psi_1(J), n_1 = J \leq J_{\max},\\ &= \psi_1(J_{\max}), n_1 > J_{\max}.\end{aligned}$$

The aggregate output rate $\psi_1(n_1)$ takes now the form displayed in Figure 4 while the state-transition diagram is given by Figure 5.

4.4 The System Long-Run Equilibrium

The local equilibrium of the aggregate is preserved in the long-run dynamics; the entire system may thus be regarded as a set of N active user terminals creating jobs for an aggregate with service rate $\psi_1(n_1)$, $n_1 = 0, \ldots, N$.

Let $\pi_2(n_1 \mid N)$, $n_1 = 0, \ldots, N$, be the long-run equilibrium probability of n_1 jobs waiting or multiprogrammed in the system; this probability distribution can be evaluated as the steady-state distribution of the congestion in a queueing system with finite population N, congestion-dependent service rate $\psi_1(n_1)$, and input rate $\lambda(N - n_1)$. By means of relations (1), long-run and unconditional distributions of the number of jobs in ready and suspended states may then be derived.

4.5 Instabilities

A particular property of the aggregate is that both its input and its output rates are functions of the number of jobs in the system. The general shape of these rates is displayed in Figure 6.

One observes that, depending on the relative values of the various parameters, there may be at most three congestion values n_A, n_B, n_C for which the input rate equates the output rate, viz. which are solutions of $\lambda(N - n_1) = \psi_1(n_1)$. It is simple to show that *n_A and n_C are stable congestions* around which the system is inclined to come into equilibrium. Let us consider the system at some instant when the congestion is in the vicinity of n_A. An increment $+\Delta n_1$ of the congestion will cause the service rate to exceed the input rate. This excess will tend to reduce the congestion to its original value n_A. Likewise a decrement $-\Delta n_1$ causes the input rate to exceed the output rate, compelling the congestion to re-increase. The same reasoning applies to congestion n_C.

Inversely, a similar argumentation indicates that in the vicinity of n_B the congestion variations are reinforced instead of deadened by the alterations they cause in the output to input rate ratio; n_B is an *unstable congestion*, a state the system will always be inclined to leave in favor of stable congestions in the vicinity of n_A or n_C.

For given λ, $J_{\max}$ and $\psi_1(n_1)$, intersection B exists for values of N between the bounds $N^{\dagger}_{J_{\max}}$ and $N^{\dagger\dagger}_{J_{\max}}$ defined graphically on Figure 6 (similar bounds could be defined as well for λ and $J_{\max}$). Between these values of N the transient behavior of the congestion will be unceasing oscillation between values in the vicinity of

n_A and values in the vicinity of n_C. This region will be characterized by a relatively large *dispersion* of the average congestion and response time of the system. This has been in part experimentally verified by a simulation study [1] of the system ESOPE.

4.6 Saturation

The mean congestion in the system, viz. the mean number of jobs waiting or multiprogrammed, is given by

$$E = \sum_{n_1=1}^{N} n_1 \times \pi_2(n_1 \mid N).$$

The considerations of Section 4.5 give an intuitive understanding of the variations of E as a function of $N, 0 < N < \infty$:

(a) For $N < N^{\dagger}_{J\max}$, only intersection A exists. The mean congestion will therefore remain in the vicinity of the stable value n_A, and, like n_A, will increase less rapidly than N.

(b) As soon as N exceeds $N^{\dagger}_{J\max}$ intersections B and C exist as well, and the mean congestion must reach abruptly an average over the range of values between the two stable values n_A and n_C.

(c) For $N > N^{\dagger\dagger}_{J\max}$ only intersection C exists; as $N \to \infty$ the system tends to behave as a system with constant service rate $\psi_1(J_{\max})$ and the mean congestion tends to increase linearly with N as does the stable value n_C. The smaller $\psi_1(J_{\max})$ is, the closer n_C to N is, obviously.

This is illustrated by Figure 7, taken from [4], where E has been plotted in function of N, for various values of $J_{\max}$, the other system parameters being given typical constant values and being such that $J_{\text{opt}} = 3$.

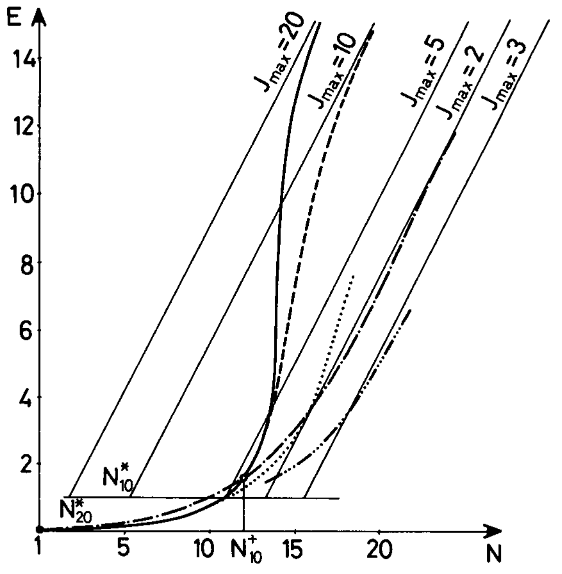
For N large, saturation sets in and E follows an asymptote of slope 1, each additional active user terminal contributing for one more unit to the congestion. One can verify that each asymptote intersects the line $E = 1$ at an abcissa $N^{*}_{J\max}$ which is precisely the saturation point defined by Kleinrock [10] for finite population models with *state independent* service rate, when this rate is taken equal to $\psi_1(J_{\max})$. This saturation point is equal to the ratio of the mean cycle time of a job when $N = 1$ (i.e. the sum of the mean job execution time and the mean terminal reaction time) to the mean job execution time. That is, here:

$$N^{*}_{J\max} = \frac{\psi_1(J_{\max})^{-1} + \lambda^{-1}}{\psi_1(J_{\max})^{-1}};$$

this is, in fact, the maximum number of active users which could possibly be serviced without queueing delays in such a system.

But, while in a model with constant service rate E approaches these asymptotes by excess [10] as $N \to \infty$, they are here (Figure 7) approached by defect. This difference results from the behavior of E for $N < N^{\dagger}_{J\max}$ (cf (a) above). $N^{*}_{J\max}$ is thus too pessimistic a definition of saturation for multiprogramming systems with *congestion-dependent* service rate of the type $\psi_1(n_1)$.

Fig. 7.



The value $N^{\dagger}_{J\max}$ provides us with a more adequate definition of saturation for such systems. Indeed, the mean congestion E increases with N with a slope which remains less than 1 for $N < N^{\dagger}_{J\max}$ and greater than 1 for $N^{\dagger}_{J\max} < N < \infty$ (this remaining true, but to a lesser extent when $J_{\max}$ is taken equal to J_{opt}). Moreover, the sharp nonlinearities in the vicinity of $N^{\dagger}_{J\max}$ are the consequence of some kind of *avalanche-like* effect which takes place as soon as the congestion exceeds n_B (which implies that intersection B exists): in this region an increase of the congestion can result in an increase of the positive difference between the input and the output rate which, in turn, accentuates the initial congestion increase. The maximal degree of multiprogramming $J_{\max}$ acts as a barrier against this avalanche-like effect: the system service rate never decreases below $\psi_1(J_{\max})$, no matter how great the congestion becomes.

This avalanche-like effect is of course due to the extreme sensitivity of the processor efficiency $(1 - \pi_1(0 \mid n_1))$, when the page transfer rate is slow, to the increase of the page fault rate when the degree of multiprogramming is allowed to exceed J_{opt}. But the above considerations show that this phenomenon, known as thrashing [6], must be related to the variations of the input and output rate to explain completely the sudden performance deteriorations of systems in which there is a feedback of the service rate upon the input load.

5. Conclusion

This analysis of the influence of program behavior upon processor efficiency, via this feedback effect upon the input load which is inherent in multiprogramming systems, would have been much less easy without the distinction of short- and long-term dynamics in nearly decomposable systems. This analysis is an example of how this distinction can be used to dissect models of computing systems into subsystems which can be (i) evaluated separately and (ii) represented by a few aggregative variables whose interactions can be analyzed at a higher level of aggregation. The degree of approximation necessitated by this approach remains known and is probably the price we have to pay to evaluate complex systems.

Acknowledgments. I am indebted to Professors H.A. Simon and D.L. Parnas, Carnegie-Mellon University, for their encouragement and many valuable suggestions; to J. Georges, MBLE Research Laboratory, because a great part of this research is the continuation of some earlier work [2] we did together; and to R. Vantilborgh, MBLE Research Laboratory, and to Professor P.J. Denning, Purdue University, whose constructive criticisms contributed to the improvement of this paper.

Received May 1973; revised May 1974

References

1. Betourne, C., and Krakowiack, S. Simulation de l'Allocation de Ressources dans un Système Conversationnel à mémoire virtuelle paginée. Proc. Congrès AFCET, Grenoble, France, Nov. 1972.
2. Buzen, J.P. Computational algorithms for closed queueing networks with exponential servers. *Comm. ACM 16*, 9 (Sept. 1973), 527-531.
3. Courtois, P.J., and Georges, J. An evaluation of the stationary behavior of computations in multiprogramming computer systems. Proc. ACM Int. Comput. Symp., Bonn, Germany, 1970, vol. 1, pp. 98-115.
4. Courtois, P.J. On the near-complete-decomposability of networks of queues and of stochastic models of multiprogramming computing systems. Scientif. Rep. CMU-CS-72-11, Carnegie-Mellon U., Nov. 1971.
5. Courtois, P.J. Error analysis in nearly decomposable stochastic systems. MBLE Rep. R214, Mar. 1973. To be published in *Econometrica* (Mar. 1975).
6. Denning, P.J. Thrashing; its causes and prevention. Proc. AFIPS 1968 FJCC, vol. 33, AFIPS Press, Montvale, N.J., pp. 915-922.
7. Dijkstra, E.W. The structure of the "THE" multiprogramming system. *Comm. ACM 11*, 5 (May 1968), 341-346.
8. Dijkstra, E.W. Hierarchical ordering of sequential processes. *Acta Informatica 1*, 2 (1971), 115-138.
9. Jackson, J.R. Jobshop-like queueing systems. *Man. Sci. 9*, 1 (Oct. 1963), 131-142.
10. Kleinrock, L. Certain analytic results for time shared processors. Proc. IFIP 68, North-Holland Pub. Co., Amsterdam, 1969, vol. 2, pp. 838-845.
11. Little, J.D.C. A proof for the queueing formula $L = \lambda W$. *Oper. Res. 9* (1961), 383-387.
12. Muntz, R., and Baskett, F. Open, closed, and mixed networks of queues with different classes of customers. Tech. Rep. N 33, Digital Syst. Lab., Stanford U., Aug. 1972.
13. Parnas, D.L., and Darringer, J.A. SODAS and a methodology for system design. Proc. AFIPS 1967 FJCC, vol. 31, AFIPS Press, Montvale, N.J., pp. 449-474.
14. Simon, H.A., and Ando, A. Aggregation of variables in dynamic systems. *Econometrica 29*, 2 (Apr. 1961), 111-138.
15. Smith, J.L. Multiprogramming under a page on demand strategy. *Comm. ACM 10*, 10 (Oct. 1967), 636-646.
16. Vantilborgh, H. On random partially preloaded page replacement algorithms. MBLE Rep. R202, Sept. 1972.
17. Zurcher, F.W., and Randell, B. Iterative multilevel modelling. A methodology for computer system design. Proc. IFIP 68 Cong., North-Holland Pub. Co., Amsterdam, 1969, vol. 2, pp. 867-871.

Operating Systems — R. Stockton Gaines, Editor

A Large Semaphore Based Operating System

Søren Lauesen
Nordisk Brown Boveri, Copenhagen

The paper describes the internal structure of a large operating system as a set of cooperating sequential processes. The processes synchronize by means of semaphores and extended semaphores (queue semaphores). The number of parallel processes is carefully justified, and the various semaphore constructions are explained. The system is proved to be free of "deadly embrace" (deadlock). The design principle is an alternative to Dijkstra's hierarchical structuring of operating systems. The project management and the performance are discussed, too. The operating system is the first large one using the RC 4000 multiprogramming system.

Key Words and Phrases: cooperating processes, operating system, semaphores, semaphore applications, queue semaphores, deadlock, deadly embrace, hierarchical structuring, multiprogramming, operating system structure, asynchronous structuring, buffering, parallel processes, synchronizing primitives, reentrant code, RC 4000, project management, time schedule, debugging, project planning, project scheduling, reliability, program proving, coroutines, correctness, program maintenance, software paging

CR Categories: 4.30, 4.31, 4.32, 4.42, 4.43, 5.24

Author's address until October 1975: UNDP, P.O. Box 1423, Accra, Ghana. Permanent address: Nordisk Brown Boveri, Vester Farimagsgade 7, DK-1606 Copenhagen V, Denmark.