Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

ON PARALLEL PROCESSING SYSTEMS: AMDAHL'S LAW GENERALIZED AND SOME RESULTS ON OPTIMAL DESIGN

Leonard Kleinrock
Jau-Hsiung Huang

October 1988
CSD-880083

On Parallel Processing Systems:

Amdahl's Law Generalized and Some Results on Optimal Design

Leonard Kleinrock and Jau-Hsiung Huang

Computer Science Department
University of California, Los Angeles

Abstract - We model a job in a parallel processing system as a concatenation of stages each of which requires a certain integral number of processors for a certain interval of time. With this model, we derive the speedup of the system for two cases: systems with no arrivals and systems with arrivals. In the case with no arrivals, our speedup result is a generalization of Amdahl's Law. We extend the notion of "*power*" (the simplest definition is $power = \frac{throughput}{response\ time}$) as previously applied to general queueing and computer-communication systems to our case of parallel processing systems. With this model, and with this definition of power, we are able to find the optimal system operating point (i.e., input rate of jobs) and the optimal number of processors to use in the parallel processing system such that power is maximized. Many of the results for the case of arrivals are the same as for the case of no arrivals. A familiar and intuitively pleasing result is obtained which states that the average number of jobs in the system with arrivals equals unity when power is maximized.

We also model a job in a way such that the number of processors required by a job is a continuous variable that changes continuously over time. The same performance indices and parameters studied in the first model are also evaluated for this continuous model. These results can be very helpful in designing a parallel processing system.

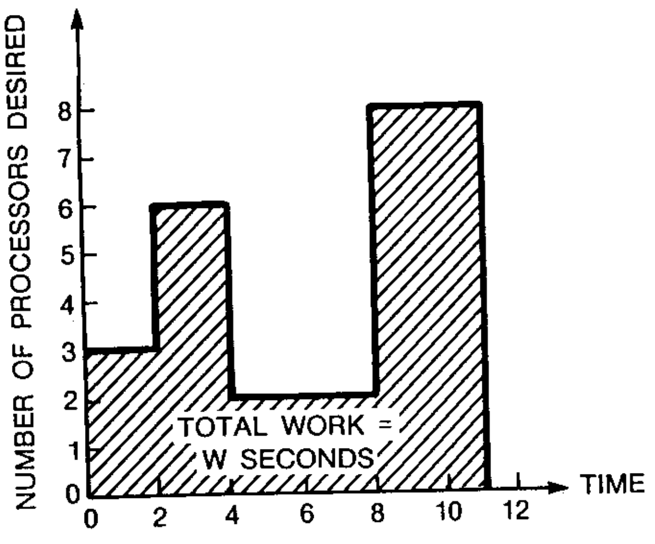
Index Terms - processor efficiency, system utilization, power, speedup, Amdahl's Law, parallel processing, multiprocessing, optimal design.

I. INTRODUCTION

As parallel computing systems proliferate, the need for effective performance evaluation techniques becomes ever more important. In this paper, we study some fundamental performance indices, maximize these indices, and find the optimal values of some design parameters. The performance indices studied in this paper are *processing time speedup*, *response time speedup*, *processor utilization*, *mean service time*, *mean response time*, and *power*. An expression for each index is given in this paper. Furthermore, by optimizing power, we are able to find λ^* (the optimal input rate of jobs) and P^* (the optimal number of processors to use) given the job specification.

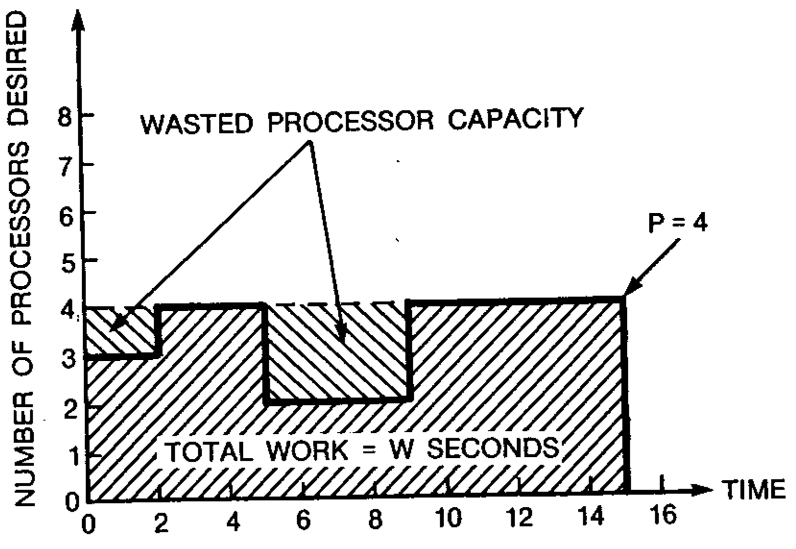
We model a parallel processing system as a system with a single queue. Only one job can be admitted into service at a time, following a FCFS discipline, while the others wait in the queue. While in service, the system provides a maximum of P parallel processors to work on the job. A job in such a system is modeled as a concatenation of stages where the number of processors required in each stage can be different. If, for some stage, the job in service requires fewer processors than the system provides, then the job will simply use all that it needs and the other processors will be idle for that stage. If, for some other stage, the job in service requires more processors than the

This research is sponsored by the Defense Advanced Research Projects Agency, Department of Defense (Contract number MDA903-87-C-0663).



a.

Processor profile when an unlimited number of processors are avaliable.



b.

Processor profile when only P processors are available.

FIGURE 1.

system provides, then it will use all the processors in the system (processor sharing) for an extended time such that the total work served in that stage is conserved. An example is given in Figure 1 in which the total processing work required by a job is W seconds.

FIGURE 1 GOES HERE.

For such a parallel processing system, there are two performance measures which compete with each other: *processor efficiency* and *mean response time*. That is, by raising the processor efficiency of the system (by reducing the number of processors), which is desirable, the mean response time will also be raised, which is not desirable. Similarly, by lowering the mean response time, the processor efficiency of the system will also be lowered. In this paper, these two performance measures are combined into a single measure, known as *power*, which increases by either lowering the mean response time or by raising the processor efficiency of the system.

Power, studied in [6] and [7], was defined in [7] as $\frac{\rho}{T/\bar{x}}$ in a general queueing system where ρ is defined as the system utilization, T is defined as the mean response time, and $\bar{x}$ is defined as the average service time. With this measure we see that an increase in system utilization (ρ) or a decrease in response time (T) increases the power. (Note that this normalized definition is such that $0 \leq \rho < 1$ and $1 \leq T/\bar{x}$ and so $0 \leq power < 1$.) The symbol * will be used to denote variables which are optimized respect to power. In [7], it was found that for any M/G/1 queueing system [5], power is maximized when $\bar{N}^* = 1$, where $\bar{N} =$ the average number of jobs in the system. This result says that an M/G/1 system has maximum power when, on the average, there is only one job in the system. This result is intuitively pleasing since it corresponds to our deterministic reasoning that the proper operating point for a single server system is exactly when only one job is being served in the system and no others are waiting for service at the same time. In this paper, our results also correspond to this deterministic reasoning when power is maximized with respect to the job arrival rate (λ).

An extensive study of power applied to computer networks is given in [3]. In [8] a system is studied where a job has a deterministic service requirement and has a processor requirement which changes during its execution time. In that model, a job requires a linearly increasing number of processors until half of the work of the job is finished; then the job needs a linearly decreasing number of processors until the other half of the job is done. Only one job is in the system; no arrivals are allowed and hence queueing effects are not considered. It is assumed in [8] that whenever the number of required processors is greater than the number of available processors in the system, the job will use all the available processors by elongating the service time under the constraint that the overall work is conserved. In [8], power was defined in the unnormalized form (from [7]) as $\frac{u(P)}{\overline{x}(P)}$, where $u(P)$ is defined as the processor efficiency during the service time of the job given there are P processors in the system and $\overline{x}(P)$ is defined as the mean service time of a job given P processors in the system. This is the appropriate definition of power if one considers the performance when only one job is to be processed. It was shown that in order to maximize power, the optimal number of processors to use in the system is approximately 60 % of the maximum number of processors required by the job when the job has a "linear profile" - see Example 1 following Theorem 3 below. This result was achieved by numerically solving a 5th order polynomial equation. In this paper we are able to obtain this result analytically for a more general case.

Speedup is a performance measure which is used to describe how much faster a job can be processed using multiple processors, as opposed to using a single processor. In this paper, we assume there are P processors in the system and we define the following notation:

$\rho \triangleq$ the system utilization, i.e., the fraction of time when there is at least one job in the system.
$\overline{x}(P) \triangleq$ the mean service time of a job given P processors in the system
$T_1(1,\rho) \triangleq$ the mean response time of a queueing system with a single processor at system utilization ρ.
$T_1(P,\rho) \triangleq$ the mean response time of a queueing system with P processors at system utilization ρ.

Note the difference between $u(P)$, which is the average *processor* efficiency given P processors, and ρ, which is the average *system* utilization. Whenever there is a job in the system, the system utilization is "1" but the processor efficiency need not be "1" since there may be some idle processors in the system because the job in service does not require all the processors. Hence, the system utilization is always greater than or equal to the processor efficiency. (Note that $u(1) = \rho$ for a single processor queueing system.)

With these definitions, we define the *processing time speedup* with P processors, denoted by $S_p(P)$, to be

$$S_p(P) = \frac{\overline{x}(1)}{\overline{x}(P)} \tag{1}$$

and we define the *response time speedup* with P processors at system utilization ρ, denoted by $S_r(P,\rho)$, to be

$$S_r(P,\rho) = \frac{T_1(1,\rho)}{T_1(P,\rho)} \tag{2}$$

Two cases are considered in this paper. Case one allows no arrivals. That is, there is only one job in the system to receive service. Case two allows arrivals at a rate λ and so queueing effects are considered. The processing time speedup is the appropriate measure to use when no arrivals are allowed in the system (i.e., only one job is to be processed) and the response time speedup is the appropriate measure when job arrivals are allowed. We define the following notation which is used, finally, to define *power*.

$u_1(P) \triangleq$ the average processor efficiency given there are P processors in the system (case of no arrivals).

$u_2(\lambda,P) \triangleq$ the average processor efficiency given the job arrival rate λ and P processors in the system.
$T_2(\lambda,P) \triangleq$ the mean response time (i.e., service time + queueing time) given the job arrival rate λ and P processors in the system [+].
$\Pi_1^{(r)}(P) \triangleq$ power (with parameter r) given P processors and no arrivals to the system.
$\Pi_2^{(r)}(\lambda,P) \triangleq$ power (with parameter r) given the job arrival rate λ and P processors in the system.

(i) If no arrivals are allowed, we define power as:

$$\Pi_1^{(1)}(P) = \frac{u_1(P)}{\bar{x}(P)} \tag{3}$$

Clearly, power will increase by either raising the processor efficiency or by lowering the mean service time. With the definition in (3), a more general definition of power (as originally studied in [7]) is given as:

$$\Pi_1^{(r)}(P) = \frac{[u_1(P)]^r}{\bar{x}(P)} \tag{4}$$

where r is a positive real number. With this generalization, we have the freedom to favor the processor efficiency more heavily than the service time by simply increasing the value of the parameter r.

(ii) If arrivals are allowed, we define power in a similar way as in (3) except that the mean service time is replaced by the mean response time since queueing will occur in a system with arrivals:

$$\Pi_2^{(1)}(\lambda, P) = \frac{u_2(\lambda,P)}{T_2(\lambda,P)} \tag{5}$$

and the generalization of (5) is given as

$$\Pi_2^{(r)}(\lambda, P) = \frac{[u_2(\lambda,P)]^r}{T_2(\lambda,P)} \tag{6}$$

where r is a positive real number.

With these definitions of power, we find the optimal number of processors to use in a parallel processing system such that power is maximized. Furthermore, if we allow arrivals, we also find the optimal system operating point (i.e., input rate of jobs).

This paper is organized as follows. In Section II we present two models of a job: a discrete model and a continuous model. In Section III we solve the case when no arrivals are allowed in the system. In this case, we find the processing time speedup of the system given P processors. We also find P^*, the number of processors which maximizes power. In Section IV we solve the case when arrivals are allowed in the system. In this case, we find the response time speedup of the system given P processors. We also find λ^* and P^* which maximize power. One interesting result we get is that the P^* for systems with no arrivals and the P^* for systems with arrivals are equal when power is maximized; this provides a great simplification in design.

II. MODELS OF JOBS

A. A Discrete Job Model

[+] $T_1(P,\rho)$ and $T_2(\lambda,P)$ are, in fact, the same function. We distinguish them to explicitly show the value of ρ in $T_1(P,\rho)$ and the value of λ in $T_2(\lambda,P)$. In all cases, we have $\rho = \lambda\bar{x}(P)$.

We model a job as containing a total of $\tilde{W}$ [illegible] Non-overlapping subsets of these tasks are collected into stages, and these stages are processed one at a time. [illegible] a random variable with mean W and coefficient of variation c_W [+]. We assume the service time distribution [illegible] each task is deterministic such that each task requires one second of work. A job is described by specifying W and c_W along with two other vectors. The first vector is called the *fraction* vector, $\vec{f}'$, and the second vector is called the *processor* vector, $\vec{P}'$. We denote the fraction vector ($\vec{f}'$) and the processor vector ($\vec{P}'$) as:

$$\vec{f}' = [\, f_1', f_2', f_3', \ldots\ldots, f_{n'}' \,]$$

$$\vec{P}' = [\, P_1', P_2', P_3', \ldots\ldots, P_{n'}' \,]$$

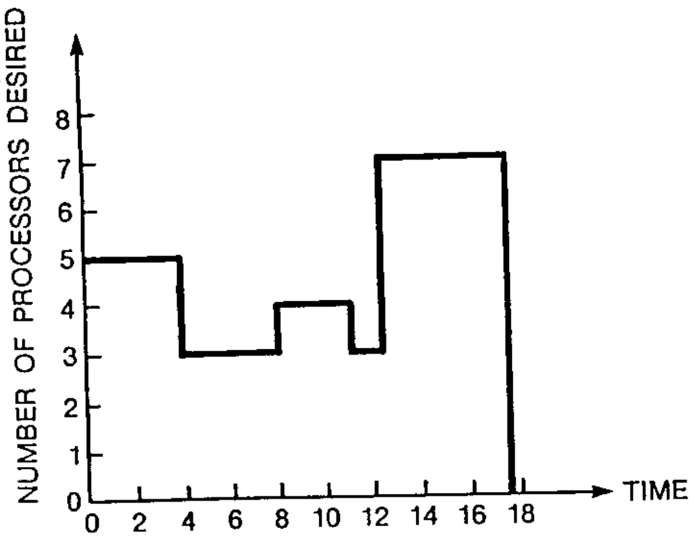
where n' is the number of stages in a job. Thus, the i^{th} stage has an f_i' and a P_i' associated with it. The meaning of $\vec{f}'$ and $\vec{P}'$ can be described as follows: a fraction f_i' of the total tasks in a job can use P_i' processors to concurrently process these tasks. From this definition, it is clear that $\sum_{i=1}^{n'} f_i' = 1$. Thus, for example, if stage i contains 10 tasks and $W = 80$, then $f_i' = 1/8$; moreover, if $P_i' = 5$ (and $P \geq 5$), then it will take 2 seconds to complete stage i.

For the convenience of computation, we can rearrange the elements in $\vec{f}'$ and $\vec{P}'$ as follows in such a way that neither the mean response time nor the processor efficiency of the system are changed. $\vec{P}'$ is rearranged so that its elements are increasing; this new vector will be denoted by $\vec{P} = [\, P_1, P_2, \ldots\ldots, P_n \,]$ such that $P_{i-1} < P_i$ in $\vec{P}$ and $\vec{f}'$ is similarly rearranged and relabeled as $\vec{f}$ accordingly. By so doing, we may merge several stages with the same P_i''s into one stage simply by adding all the corresponding f_i''s into one f_i. Since the system admits only one job into service at a time, it can easily be shown that this rearrangement does not affect the performance at all. An example is shown in Figure 2.

FIGURE 2 GOES HERE.

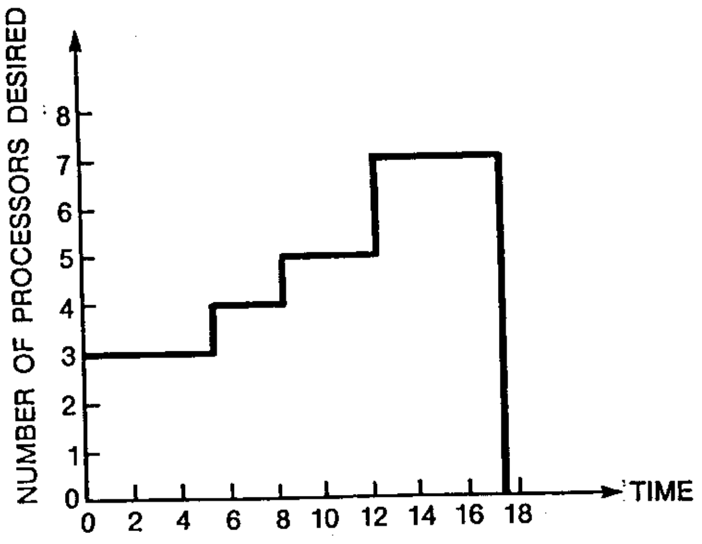
B. A Continuous Job Model

[+] the coefficient of variation of a random variable is equal to its standard deviation divided by its mean.



a. Original Processor Profile

$\bar{P}' = [5,3,4,3.7]\ \bar{f}' = [\frac{1}{4}, \frac{3}{20}, \frac{3}{20}, \frac{1}{20}, \frac{2}{5}]$



b. Processor Profile after.

$\bar{P} = [3.4.5.7]\ \bar{f} = [\frac{1}{5}, \frac{3}{20}, \frac{1}{4}, \frac{2}{5}]$

FIGURE 2.

We now describe a continuous version of the above model. In this model we assume that the number of processors required by jobs is a continuous increasing function which changes continuously over time. A special model with deterministic workload per job will be described first. After that, a general model with a random workload per job will be described.

For the special case with a deterministic workload, we define $R(t) = g(t)$ to be the function which gives the number of processors required by a job at time t and t is in the range $[0, b]$ such that $R(b) = B$. For such a model, the workload (seconds of work) for each job is deterministic with value $W = \int_0^b g(t)dt$.

For the general case, we define $\tilde{R}(t) = g(\frac{t}{\tilde{K}})$ to be a random function which gives the number of processors required by the job at time t. $\tilde{K}$ is a random variable with mean K and coefficient of variation c_K and g is a function such that it has a fixed maximum value B and t is in the range $[0, \tilde{K}b]$ as shown in Figure 3. As explained in the discrete model (Section IIA), we assume $\tilde{R}(t) = g(\frac{t}{\tilde{K}})$ to be a monotonically increasing continuous function in t without loss of generality. Later we relax the strictly increasing constraint on g.

FIGURE 3 GOES HERE.

The distribution of $\tilde{K}$ affects the results given in this paper in the following simple fashion.

(i) All time variables should be multiplied by $\tilde{K}$.

(ii) All mean time variable should be multiplied by $\bar{K}$.

(iii) P^* is independent of the distribution of $\tilde{K}$.

Therefore, we assume $\bar{K} = 1$ in the remainder of this paper. In addition, c_K does affect some of our expressions and it appears explicitly in these expressions.

III. SYSTEMS WITH NO ARRIVALS

In this section, we examine the system with no arrivals. That is, there is one job in the system to be processed using P processors and no arrivals to the system follow. We wish to find the processing time speedup and to find P^* which maximizes power. For systems with no arrivals, we have

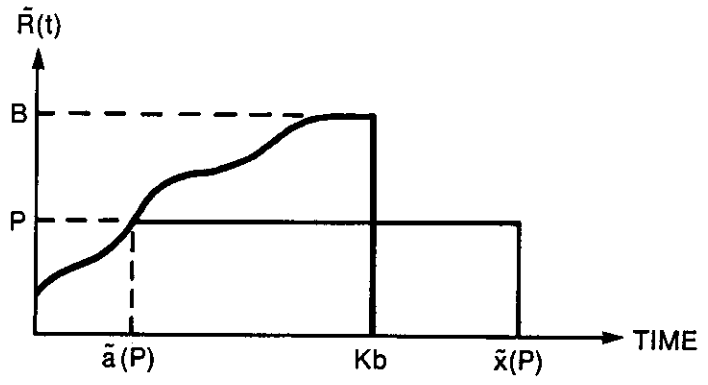


FIGURE 3. A Model with Continuously Changing Number of Processors (Random Service Requirement)

$$u_1(P) = \frac{\textit{mean workload of jobs}}{P\bar{x}(P)} \tag{7}$$

From the definitions of processing time speedup (as given in (1)) and power (as given in (3)), we see that only the mean workload of jobs, P, and $\bar{x}(P)$ are involved in the definitions. Since the mean workload of jobs, P, and $\bar{x}(P)$ are not affected by the distribution of the service requirement, we see that the distribution of the service requirement does not affect both the processing time speedup and power (only the "mean" affects these measures). Therefore, to simplify the notation, only deterministic service requirement is considered in this section. Jobs with random service requirements will have the same result as the deterministic service requirement as long as they have the same mean. We first study the continuous case and then apply the results obtained in the continuous case to the discrete case.

A. The Continuous Case

We assume $R(t)$ has the following three properties: (1) $R(t)$ is continuous and everywhere differentiable, (2) $R(t)$ is monotonically increasing, and (3) $R(t)$ is a one-to-one mapping. We define:

$a(P) \triangleq$ length of the interval from when a job first begins service until it first requires more processors than the system supplies, i.e., $a(P) = min(t : R(t) > P)$ (see Figure 4).
$b \triangleq$ the service time of the job if the number of processors in the system is always greater than the number of processors required by the job (see Figure 4).
$I(P) \triangleq \int_{a(P)}^{b} R(t)dt$

FIGURE 4 GOES HERE.

Note from Figure 4 that $I(P) = [\,\bar{x}(P) - a(P)\,]P$. The processing time speedup for this system is

$$S_p(P) = \frac{\bar{x}(1)}{\bar{x}(P)} = \frac{\int_0^b R(t)dt}{a(P) + \frac{\int_{a(P)}^{b} R(t)dt}{P}} = \frac{P\int_0^b R(t)dt}{Pa(P) + \int_{a(P)}^{b} R(t)dt} \tag{8}$$

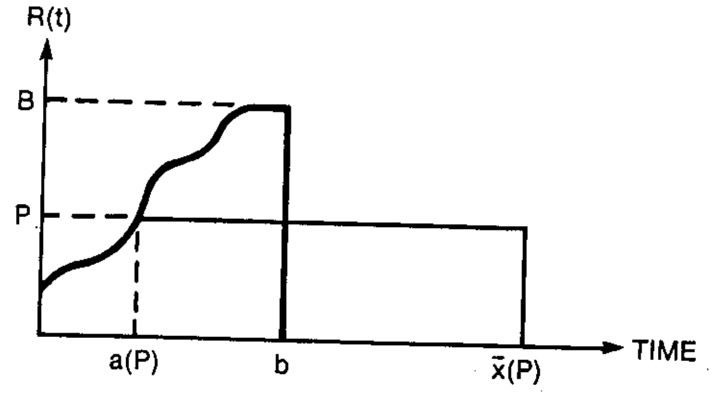


FIGURE 4. A Model with Continuously Changing Number of Processors (Deterministic Service Requirement)

THEOREM 1: Power, defined as $\frac{u_1(P)}{\bar{x}(P)}$, is maximized with respect to P when $P = P^*$, where P^* is the unique value satisfying

$$P^* = \frac{I(P^*)}{a(P^*)} \tag{9}$$

[Proof] From the definitions, we have

$$\bar{x}(P) = a(P) + \int_{a(P)}^{b} \frac{R(t)}{P} dt = a(P) + \frac{I(P)}{P} \tag{10}$$

Since $u_1(P)$ is the efficiency of the processors, we have

$$u_1(P) = \frac{\int_0^b R(t)dt}{P\bar{x}(P)} = \frac{W}{P\bar{x}(P)} \tag{11}$$

where $W = \bar{x}(1)$ = total number of seconds of work required by a job. (Note that W is independent of P.) Thus, power $\Pi_1^{(1)}(P)$ becomes

$$\Pi_1^{(1)}(P) = \frac{u_1(P)}{\bar{x}(P)} = \frac{W}{P[\bar{x}(P)]^2} \tag{12}$$

Maximizing $\Pi_1^{(1)}(P)$ with respect to P, we require

$$\frac{d}{dP} P[\bar{x}(P)]^2 = 0$$

which leads to

$$\bar{x}(P) = -2P\frac{d\bar{x}(P)}{dP} \tag{13}$$

But from (10) we have

$$\frac{d\bar{x}(P)}{dP} = \frac{da(P)}{dP} + \frac{P\frac{dI(P)}{dP} - I(P)}{P^2} \tag{14}$$

Substituting equations (10) and (14) into equation (13), we have

$$a(P) = -2P\left[\frac{da(P)}{dP} + \frac{1}{P}\frac{dI(P)}{dP}\right] + \frac{2I(P)}{P} - \frac{I(P)}{P}$$

Solving for P, we find that the optimal value of P must be such that

$$P = \frac{I(P)}{a(P)} - \frac{2P^2}{a(P)}\left[\frac{da(P)}{dP} + \frac{1}{P}\frac{dI(P)}{dP}\right] \tag{15}$$

Note that

$$\frac{dI(P)}{dP} = \frac{dI(P)}{da(P)}\frac{da(P)}{dP}$$

Moreover, since

$$I(P) = \int_{a(P)}^{b} R(t)dt$$

we have

$$\frac{dI(P)}{da(P)} = -R(a(P))$$

But $a(P)$ is such that $R(a(P)) = P$, thus $\frac{dI(P)}{da(P)} = -P$ and so $\frac{dI(P)}{dP} = -P\frac{da(P)}{dP}$; therefore

$$\frac{da(P)}{dP} + \frac{1}{P}\frac{dI(P)}{dP} = 0 \tag{16}$$

From equations (15) and (16) we see that the optimal value, P^*, is such that

$$P^* = \frac{I(P^*)}{a(P^*)}$$

It can easily be shown that $\frac{d^2}{dP^2}\Pi_1^{(1)}(P^*) < 0$; therefore, $P^* = \frac{I(P^*)}{a(P^*)}$ indeed maximizes power.

Q.E.D.

To help explain the meaning of Theorem 1, let us stage and interpret the following Corollary:

Corollary 1: Power, defined as $\frac{u_1(P)}{\overline{x}(P)}$, is maximized if and only if

$$\overline{x}(P^*) = 2a(P^*) \tag{17}$$

[Proof] Since $\frac{I(P^*)}{P^*} = a(P^*)$, we have that $\overline{x}(P^*) = a(P^*) + a(P^*) = 2a(P^*)$

Q.E.D.

To interpret this Corollary, we note from Figure 4 we know that $a(P)$ is that portion of the service time when the job has available at least as many processors as it needs. Therefore, $\overline{x}(P) = 2a(P)$ implies that the portion of the service time when there are enough processors equals the portion of the service time when there are not enough processors for its needs. Let us define $a(P)$ to be the "unextended service time" and $\frac{I(P)}{P}$ to be the "extended service time". This Corollary states that the optimal number of processors, P^*, should be selected so that the "unextended service time" equals the "extended service time". Also note that during the unextended service time, the processors are not fully utilized ($u < 1$) whereas during the extended service time, the processors are fully utilized ($u = 1$). Therefore, the time period for $u < 1$ equals the time period for $u = 1$.

THEOREM 2: Generalized power, defined as $\frac{[u_1(P)]^r}{\overline{x}(P)}$, is maximized when P^* is selected such that

$$P^* = \frac{I(P^*)}{ra(P^*)} \tag{18}$$

[Proof] The proof can easily be derived following the procedure given in the proof for Theorem 1.

Q.E.D.

Corollary 2: Generalized power, defined as $\frac{[u_1(P)]^r}{\bar{x}(P)}$, is maximized when P^* is selected such that

$$\bar{x}(P^*) = (r+1)a(P^*) \tag{19}$$

[Proof] This proof can easily be derived following the procedure given in the proof for Corollary 1.

Q.E.D.

Let us consider two examples to show the application of Theorem 2.

Example 1: If $R(t)$ is a linear function, i.e., $R(t) = \frac{B}{b}t$ for $0 \le t \le b$, and power is defined as in (4), then we have

$$P^* = \frac{B}{\sqrt{2r+1}}$$

or

$$\frac{P^*}{\textit{Maximum number of processors required}} = \frac{1}{\sqrt{2r+1}}$$

[Proof] From $R(t)$ we have

$$a(P) = \frac{b}{B}P$$

$$I(P) = \int_{\frac{bP}{B}}^{b} \frac{B}{b}t\,dt = \frac{Bb}{2} - \frac{b}{2B}P^2$$

Hence,

$$P^* = \frac{I(P^*)}{ra(P^*)} = \frac{B^2 - P^{*2}}{2rP^*}$$

Solving for P^*, we have

$$P^* = \frac{B}{\sqrt{2r+1}}$$

Note that B is the maximum number of processors required by the job, therefore,

$$\frac{P^*}{\textit{Maximum number of processors required}} = \frac{1}{\sqrt{2r+1}}$$

Q.E.D.

In this example, by setting $r = 1$, we have $\frac{P^*}{B} = \frac{1}{\sqrt{3}} \approx 58\ \%$ (where B is the maximum number of processors required). This is the case approximated in [8]; here we have the *exact* value of P^*.

Example 2: If $R(t) = \frac{B}{b^n}t^n$ for $0 \le t \le b$ and power is defined as in (4), we have

$$P^* = \frac{B}{[(n+1)r+1]^{\frac{n}{n+1}}}$$

or

$$\frac{P^*}{\textit{Maximum number of processors required}} = \frac{1}{[(n+1)r+1]^{\frac{n}{n+1}}}$$

[Proof] This proof is similar to the proof in Example 1.

Q.E.D.

In Theorem 1 we required $R(t)$ to satisfy three constraints. Unfortunately, all three constraints will be violated if we apply this result to the more natural case of an integer number of processors (as in Sections IIIB and IIIC below). In the following Theorem, we show that Corollary 1 still holds even when any or all of the constraints for $R(t)$ are violated.

THEOREM 3: Corollary 1 applies even if $R(t)$ has any or all of the following:
(1) a countable number of non-differentiable points
(2) vertical segments (such that $R(t)$ is not a one-to-one mapping)
(3) horizontal segments (such that $R(t)$ is not monotonically increasing)

[Proof] See Appendix.

Theorem 3 is important because we can use the condition in Corollary 1 (that is, $\bar{x}(P^*) = 2a(P^*)$) or in Corollary 2 (that is, $\bar{x}(P^*) = (r+1)a(P^*)$) to solve discrete cases as we now proceed to do.

B. The Discrete Case: Jobs with Two Stages

In this section, a job is modeled as consisting of W tasks of which a fraction f $(0 < f < 1)$ must be done serially (i.e., each such task can use only one processor) and of which the remaining fraction $(1-f)$ can be done concurrently with P processors, where P is the number of processors in the system. This is equivalent to $\vec{f} = [\, f,\ 1-f\,]$ and $\vec{P} = [\ 1,\ P\ ]$. Note that this is the model used by Amdahl to derive Amdahl's Law [1], which we now state:

THEOREM 4 (Amdahl's Law): The processing time speedup of this model, given P processors, is

$$S_p(P) = \frac{P}{fP + 1 - f} \tag{20}$$

[Proof] Since Wf of the tasks must be done serially, they will take Wf seconds; also since $(1-f)W$ of the tasks can be done concurrently with P processors, they will take $\frac{(1-f)W}{P}$ seconds. Therefore, the mean service time equals:

$$\bar{x}(P) = Wf + \frac{(1-f)W}{P}$$

Moreover, since a single processor working alone will take W seconds, the processing time speedup is simply W divided by $\bar{x}(P)$.

Q.E.D.

This result was first derived by Amdahl in [1] and is known as Amdahl's law. This expression implies that the processing time speedup of the system depends very much on the characteristics of the job (f) and that the processing time speedup may be much smaller than the number of processors even for relatively small f. For example, if $f = 0.1$, then one will obtain a speedup less than 10 with 1000 times the processing capacity ($P = 1000$).

THEOREM 5: Power, defined as $\frac{u_1(P)}{\bar{x}(P)}$, is maximized when the number of processors is selected to be

$$P^* = \begin{cases} 1 & \text{if } f \geq \frac{1}{2} \\ \frac{1-f}{f} & \text{if } f < \frac{1}{2} \end{cases}$$

[Proof] During the entire service time ($\bar{x}(P)$), the processing capability is $P\bar{x}(P)$; the work actually completed is simply W (since the service time for each task is 1). Hence, the processor efficiency equals

$$u_1(P) = \frac{W}{P\bar{x}(P)} = \frac{1}{Pf + 1 - f}$$

Defining power as in (3), we have

$$\Pi_1^{(1)}(P) = \frac{u_1(P)}{\bar{x}(P)} = \frac{1}{W} \cdot \frac{P}{(Pf + 1 - f)^2}$$

Optimizing $\Pi_1^{(1)}(P)$ with respect to P, it is easy to show that $P^* = \frac{1-f}{f}$. However, P^* cannot be smaller than 1 (an obvious boundary condition); hence, $P^* = 1$ if $\frac{1-f}{f} \leq 1$ (or, $f \geq \frac{1}{2}$).

Q.E.D.

The result given in Theorem 5 matches our intuition. It says that when many of the tasks have to be done serially (a large f), P^* should be small since, even with more processors, they will be wasted most of the time. On the other hand, if most of the tasks can be processed concurrently (a small f), P^* should be large since we can then achieve a higher concurrency. Figure 5 shows the curve for P^* versus f. Note that there is a sharp drop in P^* when f is small and also note that $P^* = 1$ for $f \geq \frac{1}{2}$. Therefore, the conclusion here is that when power is maximized, the concurrency can be hurt badly by a slight increase in the value of f.

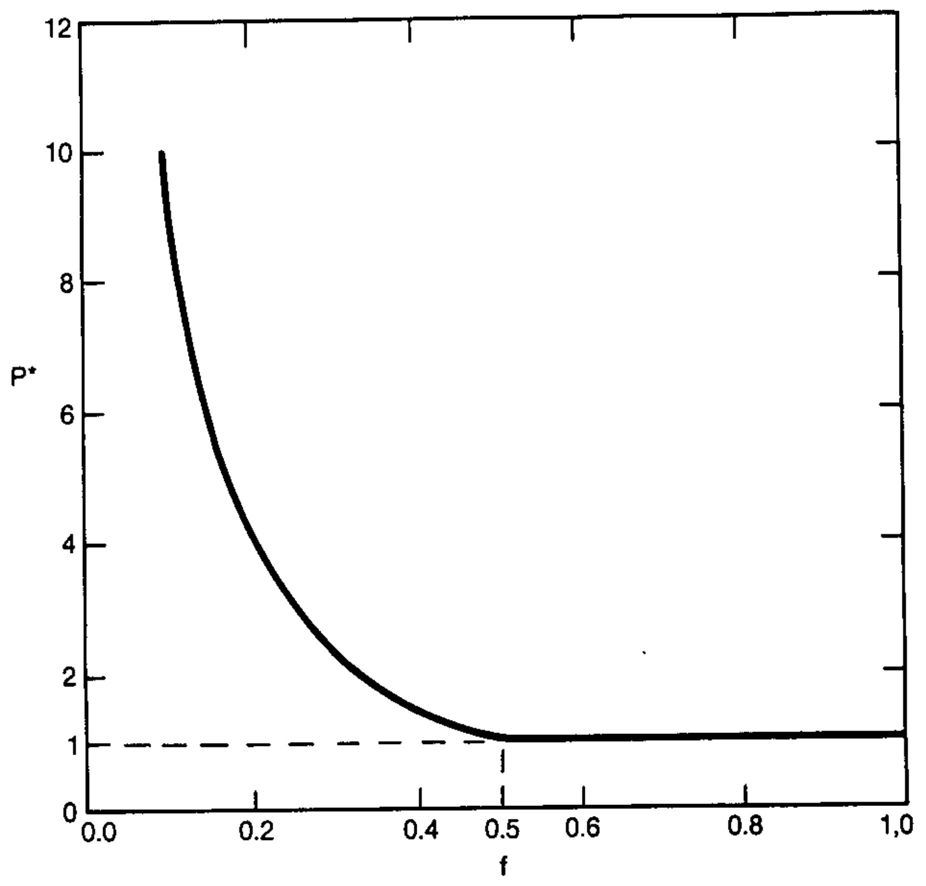


FIGURE 5.

Corollary 3: For $P^* = \frac{1-f}{f} > 1$, the interval of time when the system is working on the serial portion equals the interval of time when the system is working on the parallel portion.

[Proof]
The service time for the serial portion $= fW$. The service time for the parallel portion $= \frac{(1-f)W}{P^*} = fW$.
Q.E.D.

Corollary 3 is similar to Corollary 1 although they each apply to different environments. If we regard the service time for the serial portion as the unextended service time and regard the service time for the parallel portion as the extended service time, then Corollary 3 is exactly the same as Corollary 1.

Corollary 4: At the optimal power point,

$$S_p(P^*) = \begin{cases} 1 & \text{if } f \geq \frac{1}{2} \\ \frac{1}{2f} & \text{if } f < \frac{1}{2} \end{cases} \tag{21}$$

[Proof] From Theorem 4, we have

$$S_p(P) = \frac{P}{fP + 1 - f}$$

From Theorem 5, we have

$$P^* = \begin{cases} 1 & \text{if } f \geq \frac{1}{2} \\ \frac{1-f}{f} & \text{if } f < \frac{1}{2} \end{cases}$$

Substituting $P = P^*$ in the expression for $S_p(P)$ completes the proof.

Q.E.D.

THEOREM 6: Generalized power, defined as $\frac{[u_1(P)]^r}{\bar{x}(P)}$, is maximized when P^* is selected such that

$$P^* = \begin{cases} 1 & \text{if } f \geq \frac{1}{r+1} \\ \frac{1-f}{rf} & \text{if } f < \frac{1}{r+1} \end{cases} \tag{22}$$

[Proof] It can be shown that

$$\Pi_1^{(r)}(P) = \frac{[u_1(P)]^r}{\bar{x}(P)} = \frac{1}{W} \cdot \frac{P}{(Pf + 1 - f)^{r+1}}$$

Optimizing $\Pi_1^{(r)}(P)$ with respect to P, one finds that $P^* = \frac{1-f}{rf}$. However, P^* must not be smaller than 1; hence

$P^* = 1$ if $\frac{1-f}{rf} \le 1$ (or, $f \ge \frac{1}{r+1}$).

Q.E.D.

Corollary 5: For $P^* = \frac{1-f}{rf} > 1$, we have

$$r \cdot (\textit{service time} \text{ for } \textit{the serial portion}) = (\textit{service time} \text{ for } \textit{the parallel portion}) \tag{23}$$

[Proof]
The service time for the serial portion = fW. The service time for the parallel portion = $\frac{(1-f)W}{P^*} = r \cdot fW$.
Q.E.D.

Corollary 6: At the optimal power point,

$$S_p(P^*) = \begin{cases} 1 & \text{if } f \ge \frac{1}{2} \\ \frac{1}{(r+1)f} & \text{if } f < \frac{1}{2} \end{cases} \tag{24}$$

[Proof] This proof is similar to the proof for Corollary 4.

Q.E.D.

C. The Discrete Case in General

For the general case, we assume that a job has W tasks and that the fraction vector and the processor vector (after rearrangement) are

$$\vec{f} = [\, f_1, f_2, f_3, \text{.........}, f_n]$$

$$\vec{P} = [\, P_1, P_2, P_3, \text{.........}, P_n]$$

where $P_i < P_{i+1}$ for $1 < i < n-1$ and $\sum_{i=1}^{n} f_i = 1$.

Before describing the Theorem, we need some more notation. We assume that there are P processors available in the system. We define the index "m" such that $P_{m-1} \le P < P_m$ if $P_1 \le P < P_n$; or, $m = n+1$ if $P \ge P_n$; or, $m = 1$ if $P < P_1$. Once m is determined, we define

$$\alpha = \sum_{i=1}^{m-1} \frac{f_i}{P_i} \tag{25}$$

and

$$\beta = \sum_{i=m}^{n} f_i \tag{26}$$

Note that αW is the unextended service time, whereas, $\frac{\beta W}{P}$ is the extended service time.

THEOREM 7 (Amdahl's Law Generalized): The processing time speedup for any P is given by

$$S_p(P) = \frac{P}{\alpha P + \beta} \tag{27}$$

[Proof] The mean service time is

$$\bar{x}(P) = W\left(\sum_{i=1}^{m-1} \frac{f_i}{P_i} + \frac{1}{P}\sum_{i=m}^{n} f_i\right) = W\left(\alpha + \frac{1}{P}\beta\right) \tag{28}$$

Since a single processor working alone will take W seconds to serve a job, the processing time speedup with P processors is simply W divided by $\bar{x}(P)$.
Q.E.D.

Note that this is a generalization of Amdahl's Law [1]! The maximum possible processing time speedup, denoted as $S_{p,max}$, for this model will be achieved when $P \geq P_n$, which gives $\alpha = \sum_{i=1}^{n} \frac{f_i}{P_i}$ and $\beta = 0$. Hence,

$$S_{p,max} = \frac{W}{\sum_{i=1}^{n} \frac{f_i W}{P_i}} = \frac{1}{\sum_{i=1}^{n} \frac{f_i}{P_i}} \tag{29}$$

In the following two Corollaries, we derive P^* for the discrete cases using the results from Corollary 1 and Theorem 3. A result similar to that of Corollary 7 is also obtained in [2].

Corollary 7 $^+$: Power, defined as $\frac{u_1(P)}{\bar{x}(P)}$, is maximized when P^* satisfies either one of the following two conditions:

$$(i) \qquad P^* = \frac{\beta}{\alpha} \quad \text{if} \quad P_{m-1} < P^* < P_m \tag{30}$$

or

$$(ii) \qquad P^* = P_{m-1} \quad \text{if} \quad 0 \leq \frac{\alpha}{2} - \frac{\beta}{2P_{m-1}} \leq \frac{f_{m-1}}{P_{m-1}} \tag{31}$$

[Proof] In case (i) there is no ambiguity in defining the unextended service time. Specifically, the unextended service time = αW and the extended service time = $\frac{\beta W}{P}$. From Corollary 1 and Theorem 3 we must have

$$\alpha W = \frac{\beta W}{P^*}$$

Hence,

$$P^* = \frac{\beta}{\alpha}$$

In case (ii), we do encounter an ambiguity in defining the place where the unextended service time ends (and thus the extended service time begins). In order to resolve this, we define x to be the interval in stage $m-1$ in the extended service time and the interval $t_{m-1} - x$ ($t_{m-1} = f_{m-1}W/P_{m-1}$ is the mean service time for stage $m-1$) to be the interval in stage $m-1$ in the unextended service time, as shown in Figure 6. From Corollary 1 and Theorem 3

$^+$ It is easy to show that P^* will never be greater than P_n. If $P^* > P_n$, the service time will not be improved while the processor utilization will get smaller than when $P^* = P_n$; hence, power will get smaller. A similar argument shows that P^* will never be smaller than P_1.

and assuming $P^* = P_{m-1}$, we have

$$\alpha W - x = \frac{\beta W}{P_{m-1}} + x$$

which gives us

$$x = \frac{\alpha W}{2} - \frac{\beta W}{2P_{m-1}}$$

Since $0 \le x \le t_{m-1}$, we have

$$0 \le \frac{\alpha W}{2} - \frac{\beta W}{2P_{m-1}} \le t_{m-1}$$

Since $t_{m-1} = \dfrac{f_{m-1} W}{P_{m-1}}$, we have

$$0 \le \frac{\alpha}{2} - \frac{\beta}{2P_{m-1}} \le \frac{f_{m-1}}{P_{m-1}}$$

Q.E.D.

FIGURE 6 GOES HERE.

Corollary 8: Generalized power, defined as $\dfrac{[u_1(P)]^r}{\bar{x}(P)}$, is maximized when P^* satisfies either one of the following two conditions:

$$(i) \qquad P^* = \frac{\beta}{r\alpha} \quad \text{if} \quad P_{m-1} < P^* < P_m \tag{32}$$

or

$$(ii) \qquad P^* = P_{m-1} \quad \text{if} \quad 0 \le \frac{r\alpha}{r+1} - \frac{\beta}{(r+1)P_{m-1}} \le \frac{f_{m-1}}{P_{m-1}} \tag{33}$$

[Proof] This proof is similar to the one for Theorem 8.
Q.E.D.

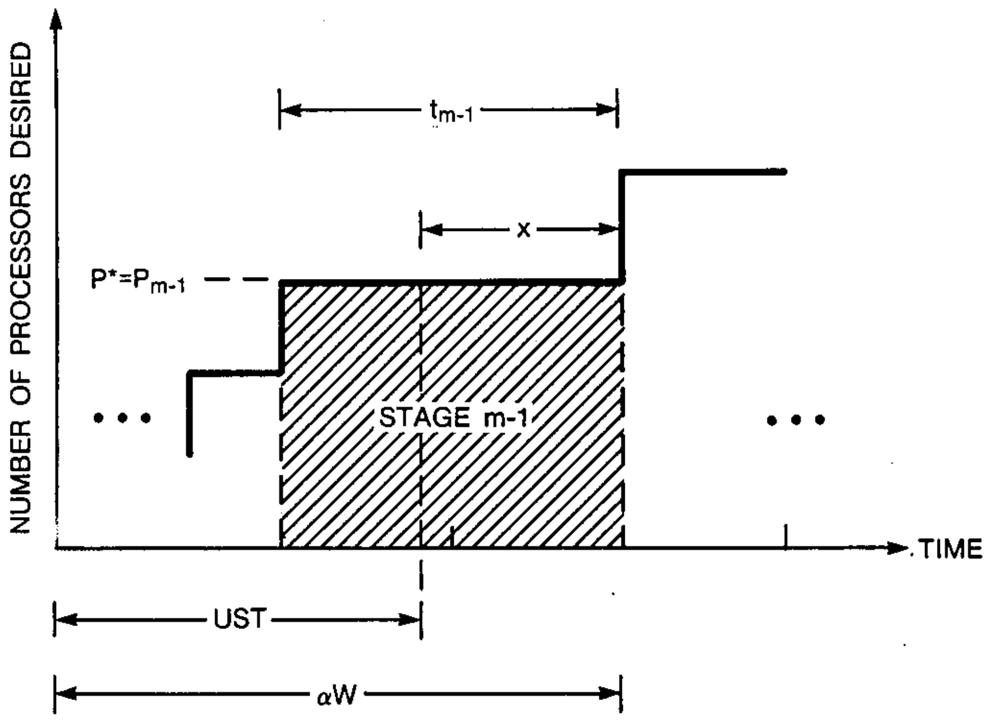


FIGURE 6. (UST = Unextended Service Time)

From Corollaries 7 and 8 we develop an iterative procedure in [4] to find P^* for any given $\vec{f}$ and $\vec{P}$. In that procedure, the number of iterations is upper bounded by $\log_2 n$, which is reasonably small.

In our results of Section IIIB and IIIC, we neglected the physical requirement that P^* be an integer. Clearly, if P^* requires to be an integer, P^* should then be rounded up or down to the nearest integer, whichever of the two has a larger value of power.

IV. SYSTEMS WITH ARRIVALS

In this section we study the case when new jobs enter the system according to a Poisson process with rate λ. The following Theorem describes a very useful property which results in many amazing results later in this section.

THEOREM 8: For all cases (continuous and discrete models), the coefficient of variation of the service time distribution (denoted as c_{x_P} when there are P processors in the system) is not a function of P. That is, for all $P \geq 1$,

$$c_{x_1} = c_{x_P} \tag{34}$$

[Proof] We define $\tilde{X}(P)$ to be the random variable representing the service time distribution when P processors are available. For the continuous model, we can show that

$$\tilde{X}(P) = \tilde{K}\left[a(P) + \int_{a(P)}^{b} \frac{g(t)}{P}dt\right] \tag{35}$$

(35) shows that $\tilde{X}(P)$ equals $\tilde{K}$ multiplied by a (deterministic) constant; since this constant multiplies both the standard deviation and the mean of $\tilde{X}(P)$, it will cancel out in their ratio (i.e., the coefficient of variation) and so

$$c_{x_P} = c_K \tag{36}$$

Hence, c_{x_P} is not a function of P, which implies $c_{x_1} = c_{x_P}$.

For the two-stage discrete case, we have

$$\tilde{X}(P) = \tilde{W}f + \frac{(1-f)\tilde{W}}{P} = \tilde{W}\left[f + \frac{(1-f)}{P}\right] \tag{37}$$

Hence, using a similar argument as above, we have

$$c_{x_P} = c_{x_1} = c_W \tag{38}$$

Similarly, for the general discrete case, we have

$$\tilde{X}(P) = \tilde{W}\alpha + \frac{\beta\tilde{W}}{P} = \tilde{W}\left[\alpha + \frac{\beta}{P}\right] \tag{39}$$

Hence, using the same argument as above, we have

$$c_{x_P} = c_{x_1} = c_W \tag{40}$$

Q.E.D.

Defining $\tilde{W}$ to be the random variable representing the work brought in by a job, we can show that

$$\tilde{W} = \int_0^{\tilde{K}b} g(\frac{t}{\tilde{K}})dt = \tilde{K}\int_0^b g(t)dt \qquad (41)$$

Since $\int_0^b g(t)dt$ is a constant, (41) shows that the work brought by a job is a random variable which has the same coefficient of variation as $\tilde{K}$.

A. Finding the Response Time Speedup

In this section we find the response time speedup for all cases. Amazingly, we discover that the response time speedup when queueing is allowed is the same as the processing time speedup when queueing is not allowed.

THEOREM 9: For all cases (continuous and discrete models), we have

$$S_r(P, \rho) = S_p(P) \qquad (42)$$

[Proof] We define ρ to be the system utilization; hence,

$$\rho = \lambda\bar{x}(P)$$

Since only one job can be admitted into service at a time, this system can be analyzed as a single server system. Hence, we can apply results from M/G/1 theory [5] to find the average response time for this system.

$$T_1(P, \rho) = \bar{x}(P)\left[1 + \rho\frac{1 + {c_{x_P}}^2}{2(1-\rho)}\right] \qquad (43)$$

In Theorem 8 we have shown that $c_{x_1} = c_{x_P}$ for all cases; thus, we find the response time speedup as

$$S_r(P, \rho) = \frac{T_1(1, \rho)}{T_1(P, \rho)} = \frac{\bar{x}(1)}{\bar{x}(P)} = S_p(P)$$

Q.E.D.

Therefore, the response time speedup and the processing time speedup are solely determined by the job specification and P (and not affected by the system's operating point) in our models. (Another interesting model studied in [4] has a different response time speedup and processing time speedup.)

Corollary 9: For the continuous model, we have

$$S_r(P, \rho) = \frac{P\int_0^b R(t)dt}{Pa(P) + \int_{a(P)}^b R(t)dt} \qquad (44)$$

[Proof] This can easily be proved from (8) and Theorem 9.

Q.E.D.

Corollary 10: For the two-stage discrete model, we have

$$S_r(P,\ \rho) = \frac{P}{fP + 1 - f} \tag{45}$$

[Proof] This can easily be proved from Theorems 4 and 9.

Q.E.D.

Corollary 11: For the general discrete model, we have

$$S_r(P,\ \rho) = \frac{P}{\alpha P + \beta} \tag{46}$$

[Proof] This can easily be proved from Theorems 7 and 9.

Q.E.D.

B. The Optimal Arrival Rate

In this section, we find the optimal operating point (λ^*) for both the discrete case and the continuous case. Amazingly, even though the definitions of power in this paper and in [7] are different (since $\rho \neq u_2(\lambda,P)$), the results obtained in both papers are the same. Therefore, all the deterministic reasoning given in [7] also applies in this paper.

THEOREM 10: Power, defined as $\frac{u_2(\lambda,P)}{T_2(\lambda,P)}$, is maximized when $\lambda = \lambda^*$ (for a given P) such that

$$\lambda^* = \frac{2}{2 + \sqrt{2 + 2c_{x_P}^2}} \cdot \frac{1}{\bar{x}(P)} \tag{47}$$

[Proof]:

$$u_2(\lambda,P) = \frac{\lambda W}{P}$$

From M/G/1 we have

$$T_2(\lambda,P) = \bar{x}(P)\left[1 + \rho\frac{1 + c_{x_P}{}^2}{2(1-\rho)}\right] = \bar{x}(P)\left[\frac{2 + (c_{x_P}^2 - 1)\lambda\bar{x}(P)}{2(1 - \lambda\bar{x}(P))}\right]$$

Defining power as in (5), we have

$$\Pi_2^{(1)}(\lambda,\ P) = \frac{u_2(\lambda,P)}{T_2(\lambda,P)} = \frac{\lambda W}{P} \cdot \frac{2(1 - \lambda\bar{x}(P))}{2 + (c_{x_P}{}^2 - 1)\lambda\bar{x}(P)} \cdot \frac{1}{\bar{x}(P)}$$

Maximizing power with respect to λ, we have

$$\lambda^* = \frac{2}{2 + \sqrt{2 + 2c_{x_P}^2}} \cdot \frac{1}{\bar{x}(P)}$$

Q.E.D.

Corollary 12: When power is maximized with respect to λ,

$$\rho^* = \frac{2}{2+\sqrt{2+2c_{x_P}^2}} \tag{48}$$

and

$$\overline{N}^* = 1 \tag{49}$$

[Proof] From Theorem 10 we trivially show that

$$\rho^* = \lambda^* \overline{x}(P) = \frac{2}{2+\sqrt{2+2c_{x_P}^2}}$$

Using Little's result [9], it is easy to show that

$$\overline{N}^* = \lambda^* T_2(\lambda^*, P) = 1$$

Q.E.D.

The result given in (49) is intriguing. Indeed, $\overline{N}^* = 1$ corresponds to the same deterministic reasoning given in [7] and which is described in our Introduction.

THEOREM 11: Generalized power, defined as $\frac{[u_2(\lambda,P)]^r}{T_2(\lambda,P)}$, is maximized when $\lambda = \lambda^*$ (for a given P) such that

$$\lambda^* = \frac{4r}{(-c_{x_P}^2+3)r + (c_{x_P}^2+1) + \sqrt{(c_{x_P}^4+2c_{x_P}^2+1)r^2 + 2(-c_{x_P}^4+2c_{x_P}^2+3)r + (c_{x_P}^2+1)^2}} \cdot \frac{1}{\overline{x}(P)} \tag{50}$$

[Proof] This proof is similar to the proof for Theorem 10.

Corollary 13: When power is maximized with respect to λ, then

$$\rho^* = \frac{4r}{(-c_{x_P}^2+3)r + (c_{x_P}^2+1) + \sqrt{(c_{x_P}^4+2c_{x_P}^2+1)r^2 + 2(-c_{x_P}^4+2c_{x_P}^2+3)r + (c_{x_P}^2+1)^2}}$$

and

$$\overline{N}^* = \frac{2r\left[(1+c_{x_P}^2)r + b(r) + (1+c_{x_P}^2)\right]}{(c_{x_P}^4-1)r^2 + 2(2-c_W^2)(1+c_{x_P}^2)r + \left[(1-c_{x_P}^2)r + (1+c_{x_P}^2)\right]b(r) + (c_{x_P}^2+1)^2}$$

where

$$b(r) = \sqrt{(c_{x_P}^4+2c_{x_P}^2+1)r^2 + 2(-c_{x_P}^4+2c_{x_P}^2+3)r + (1+c_{x_P}^2)^2}$$

If $r >> 1$, we have

$$\lim_{r\to\infty} \rho^* = \frac{r}{r+1}$$

and

$$\lim_{r\to\infty} \overline{N}^* = \frac{(1+c_{x_P}^2)r}{2}$$

Note that the results in Theorem 11 and Corollary 13 are the same as in [7].

C. Finding the optimal number of Processors (P^*)

In this section, we first study the relationship between $\Pi_1^{(1)}(P)$ and $\Pi_2^{(1)}(\lambda,P)$. From the result below, we show that there are many cases in which P^* for a system with no arrivals and P^* for a system with arrivals are the same!

We may express the processor utilization, $u_2(\lambda,P)$ for systems with arrivals in terms of the processor utilization, $u_1(P)$ for systems with no arrivals as follows:

$$u_2(\lambda,P) = (\textit{processor utilization})$$

$$= (\textit{processor utilization} \mid \textit{system busy})\cdot P[\textit{system busy}] + (\textit{processor utilization} \mid \textit{system idle})\cdot P[\textit{system idle}]$$

$$= (\textit{processor utilization} \mid \textit{system busy})\cdot P[\textit{system busy}]$$

Thus we come to the simple conclusion that

$$u_2(\lambda,P) = u_1(P)\cdot\rho \tag{51}$$

Substituting $u_2(\lambda,P) = \rho u_1(P)$ into the definition of power, we find that

$$\Pi_2^{(1)}(\lambda,P) = \frac{u_2(\lambda,P)}{T_2(\lambda,P)} = \frac{\rho u_1(P)}{T_2(\lambda,P)} = \frac{\rho}{T_2(\lambda,P)/\overline{x}(P)}\cdot\frac{u_1(P)}{\overline{x}(P)}$$

Since $\dfrac{u_1(P)}{\overline{x}(P)} = \Pi_1^{(1)}(P)$ and $\dfrac{\rho}{T_2(\lambda,P)/\overline{x}(P)} = \dfrac{2\rho(1-\rho)}{2-\rho+\rho c_{x_P}^2}$ for M/G/1, we finally have

$$\Pi_2^{(1)}(\lambda,P) = \frac{2\rho(1-\rho)}{2-\rho+\rho c_{x_P}^2}\cdot\Pi_1^{(1)}(P) \tag{52}$$

Note that $\rho/[T_2(\lambda,P)/\overline{x}(P)]$ is simply the normalized power discussed in [7].

Let us now discuss the optimal number of processors, P^*. When the system is operating at the optimal operating point (λ^*), we have

$$\Pi_2^{(1)}(\lambda^*,P) = \frac{2\rho^*(1-\rho^*)}{2-\rho^*+\rho^* c_{x_P}^2}\cdot\Pi_1^{(1)}(P)$$

Note that $\dfrac{2\rho^*(1-\rho^*)}{2-\rho^*+\rho^* c_{x_P}^2}$ is only a function of c_{x_P} (since ρ^* is a function of c_{x_P} only as shown in Corollary 13) and, in particular, is not a function of P; therefore, for cases where c_{x_P} is not a function of P, then P^* for $\Pi_2^{(1)}(\lambda^*,P)$ is the same as P^* for $\Pi_1^{(1)}(P)$. That is, for c_{x_P} not a function of P, we have systems with

$$P^*(\textit{for systems with no arrivals}) = P^*(\textit{for systems with arrivals}) \tag{53}$$

For the generalized definition of power, we have

$$\Pi_2^{(r)}(\lambda,P) = \frac{[u_2(\lambda,P)]^r}{T_2(\lambda,P)} = \frac{\rho^r[u_1(P)]^r}{T_2(\lambda,P)} = \frac{\rho^r}{T_2(\lambda,P)/\overline{x}(P)}\cdot\frac{[u_1(P)]^r}{\overline{x}(P)} = \frac{2\rho^r(1-\rho)}{2-\rho+\rho c_{x_P}^2}\cdot\Pi_1^{(r)}(P) \tag{54}$$

Using the same argument as above (i.e., for c_{x_P} not a function of P), we have systems with this characteristic. In [4], another model is discussed in which c_{x_P} is indeed a function of P. In that case, a numeric procedure is required to find P^*.

Corollary 14: For the continuous model, power, as defined in (5), is maximized when P^* is chosen such that

$$P^* = \frac{I(P^*)}{a(P^*)} \tag{56}$$

and

$$\lambda^* = \frac{1}{a(P^*)(2 + \sqrt{2 + c_K^2})} \tag{57}$$

[Proof] This is easily derived from Theorems 1 and 10.
Q.E.D.

Corollary 15: For the two-stage discrete model, power, as defined in (5), is maximized when

$$\lambda^* = \frac{1}{fW(2 + \sqrt{2 + 2c_W^2})} \tag{58}$$

and the optimal number of processors is

$$P^* = \begin{cases} 1 & \text{if } f \geq \frac{1}{2} \\ \frac{1-f}{f} & \text{if } f < \frac{1}{2} \end{cases} \tag{59}$$

[Proof] This is easily derived from Theorems 5 and 10.
Q.E.D.

V. CONCLUSION

For the model which allows no arrivals, we have found the processing time speedup, ($S_p(P)$) for any P, and the optimal number of processors (P^*) which maximizes power. This $S_p(P)$ was shown to be a generalization of Amdahl's Law. For the model which allows arrivals, we have found the response time speedup ($S_r(P,\rho)$) for any P, the optimal arrival rate (λ^*) and the optimal number of processors (P^*) which maximize power. It was amazing to find that $S_p(P)$ is the same as $S_r(P,\rho)$ for the models studied in this paper. It was even more amazing to find that P^* for a system with no arrivals is the same as P^* for a system with arrivals when power is maximized. In all cases, we found that power is optimized when P^* is chosen so that the unextended service time equals the extended service time. This characteristic makes optimal design (in terms of maximizing power) easier because the same solution applies to both cases!

References

[1] Amdahl, G.M., "Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities," Proceedings of AFIPS, Vol. 30, 1967.

[2] Eager, D.L., Zahorjan, J., and Lazouska, E.D., "Speedup Versus Efficiency in Parallel Systems," Technical report 86-08-01, Department of Computer Science, University of Washington, August, 1986.

[3] Gail, H.R., "On the Optimization of Computer Network Power," Ph.D. dissertation, Computer Science Department, UCLA. September 1983.

[4] Huang, J., "On the Behavior of Algorithms in a Multiprocessing Environment," Ph.D. dissertation, Computer Science Department, UCLA, 1988.

[5] Kleinrock, L., *Queueing Systems, Vol.* 1: *Theory*, Wiley- Interscience, New York, 1975.

[6] Kleinrock, L., "On Flow Control in Computer Networks," Conference Record, International Conference on Communications 2, pp. 27.2.1 - 27.2.5, June 1978.

[7] Kleinrock, L., "Power and Deterministic Rules of Thumb for Probabilistic Problems in Computer Communications," Conference Record, International Conference on Communications, June 1979, pp. 43.1.1-43.1.10.

[8] Kung, K.C., "Concurrency in Parallel Processing Systems," Ph.D. Dissertation, Computer Science Department, UCLA, 1984.

[9] Little, J.D.C., "A Proof of the Queueing Formula $L = \lambda W$," *Operations Research*, 9, 1961. pp. 383-387.

APPENDIX (Proof of Theorem 3)

THEOREM 3: Corollary 1 applies even if $R(t)$ has
(1) a countable number of non-differentiable points
(2) vertical segments (such that $R(t)$ is not a one-to-one mapping)
(3) horizontal segments (such that $R(t)$ is not monotonically increasing)

[Proof] To prove (1), we first prove the case when the non-differentiable point is at the intersection of two straight lines. We then generalize it for all cases. To prove the case when the non-differentiable point is at the intersection of two straight lines, as shown in Figure 7(a), we assume the following (see Figure 7(a)):
(a) there is one non-differentiable corner at point D=(x_d, y_d),
(b) the work (area) under $R(t)$ beyond point D is I (i.e., $I = \int_{x_d}^{b} R(t)dt$) and $\frac{I}{y_d} = x_d$ (i.e., point $P^* = y_d$ meets the condition in Corollary 1),
(c) for some $\delta > 00$, we can find points E=(x_e, y_e) and F=(x_f, y_f) are on $R(t)$ such that the length of line segments $\overline{DE}$ = the length of line segment $\overline{DF} = \delta$.
(d) the slope of line segment $\overline{DE}$ is s_e and the slope of line segment $\overline{DF}$ is s_f such that $s_e \neq s_f$.

Using (c) and (d) we can fit a unique circle which is tangent to $\overline{DE}$ at point E and which is tangent to $\overline{DF}$ at point F.

FIGURE 7a GOES HERE.

The way to find this unique circle can easily be described as follows:
(i) at point E, draw a line (L_1) which is perpendicular to $\overline{DE}$ and at point F, draw a line (L_2) which is perpendicular to $\overline{DF}$.
(ii) Since the slopes for $\overline{DE}$ and $\overline{DF}$ are different, L_1 and L_2 will intersect at some point O.
(iii) Using point O as the center of the circle and $\overline{OE}$ ($\overline{OE} = \overline{OF}$) as the radius, we find this unique circle which passes through points E and F and is tangent to $\overline{DE}$ and $\overline{DF}$ at points E and F.

With this circle, we define a new curve $R_1(t)$ which coincides with $R(t)$ except that the line segments $\overline{FD}$ and $\overline{DE}$ are replaced by the circle between points E and F. Clearly, $R_1(t)$ meets the requirement for Corollary 1 (that is, no non-differentiable points) and hence we may apply the result in Corollary 1. Let us define P_1^* to be the optimal number of processors to maximize power for $R_1(t)$ and assume P_1^* happens at the point G=(x_g, y_g) such that $\frac{I_1}{y_g} = x_g$, where I_1 is the area under $R_1(t)$ beyond point G (i.e., $I_1 = \int_{x_g}^{b} R_1(t)dt$). Next, we must show that $\lim_{\delta \to 0} G = D$. Clearly, we have $\lim_{\delta \to 0} R_1(t) = R(t)$. Assuming $\lim_{\delta \to 0} G$ does not equal D and is to the right of D (i.e., $x_g > x_d$), then we have $I > I_1$, $x_g > x_d$, and $y_g > y_d$, (since $R(t)$ is an increasing function of t). However, we then have $x_d y_d < x_g y_g$ which implies $I < I_1$, a contradiction! Hence, point G cannot be to the right of point D. A similar argument shows that point G cannot be to the left of point D. Hence, we come to the conclusion that $\lim_{\delta \to 0} G = D$. This implies that y_d is P^* for curve $R(t)$; that is, we can apply Corollary 1 to $R(t)$ directly. If there is a countable number of such non-differentiable points, then we may apply the argument above to each such point separately.

We now prove (1) for cases when the non-differentiable point is at the intersection of two curves as shown in Figure 7(b). For this situation, we find points E=(x_e, y_e) and F=(x_f, y_f) such that $x_e = x_d + \delta$ and $x_f = x_d - \delta$. Then, we find the tangent lines at points E and F, say L_1 and L_2 respectively. If δ is small enough, L_1 and L_2 will intersect at point D'. We define a new curve $R_1(t)$ as the same as $R(t)$ except that the curve from F to D to E is replaced by the line segments $\overline{FD'}$ and $\overline{D'E}$. For $R_1(t)$, it only has one non-differentiable point at the intersection of two straight lines. In this case, we can apply Corollary 1 to $R_1(t)$ as we just proved above. Hence, by letting $\delta \to 0$, a similar technique and proof can be applied here to show that Corollary 1 also applies in this case. (1) is thus proved.

The proofs for (2) and (3) are similar to the proof for (1). Hence, only an abstract is given here without much detail. For case (2), as shown in Figure 7(c), we find points E=(x_e, y_e) and F=(x_f, y_f) such that $x_e = x_d + \delta$ and $x_f = x_d - \delta$ and we form a line segment between points E and F. We define $R_1(t)$ to be the same as $R(t)$ except that the curve between points E and F is replaced by the straight line between points E and F. Then, $R_1(t)$ meets the condition for (1) (i.e., with only non-differentiable points). Letting $\delta \to 0$, we can use a similar argument as given in the proof for (1) (i.e., $\lim_{delta \to 0} G = D$) to show that Corollary 1 applies in case (2).

Similarly, for case (3) as shown in Figure 7(d), we find points E=(x_e, y_e) and F=(x_f, y_f) such that $y_e = y_d + \delta$ and $y_f = y_d - \delta$, and we form a line segment between points E and F. We define $R_1(t)$ to be the same as $R(t)$ except that the curve between points E and F is replaced by the straight line between points E and F. Then, $R_1(t)$ meets the condition for (1) (i.e., with only non-differentiable points). Letting $\delta \to 0$, we can use a similar argument as given in the proof for (1) to show that Corollary 1 applies in case (3).

Q.E.D.

FIGURES 7b, 7c, 7d GOES HERE.

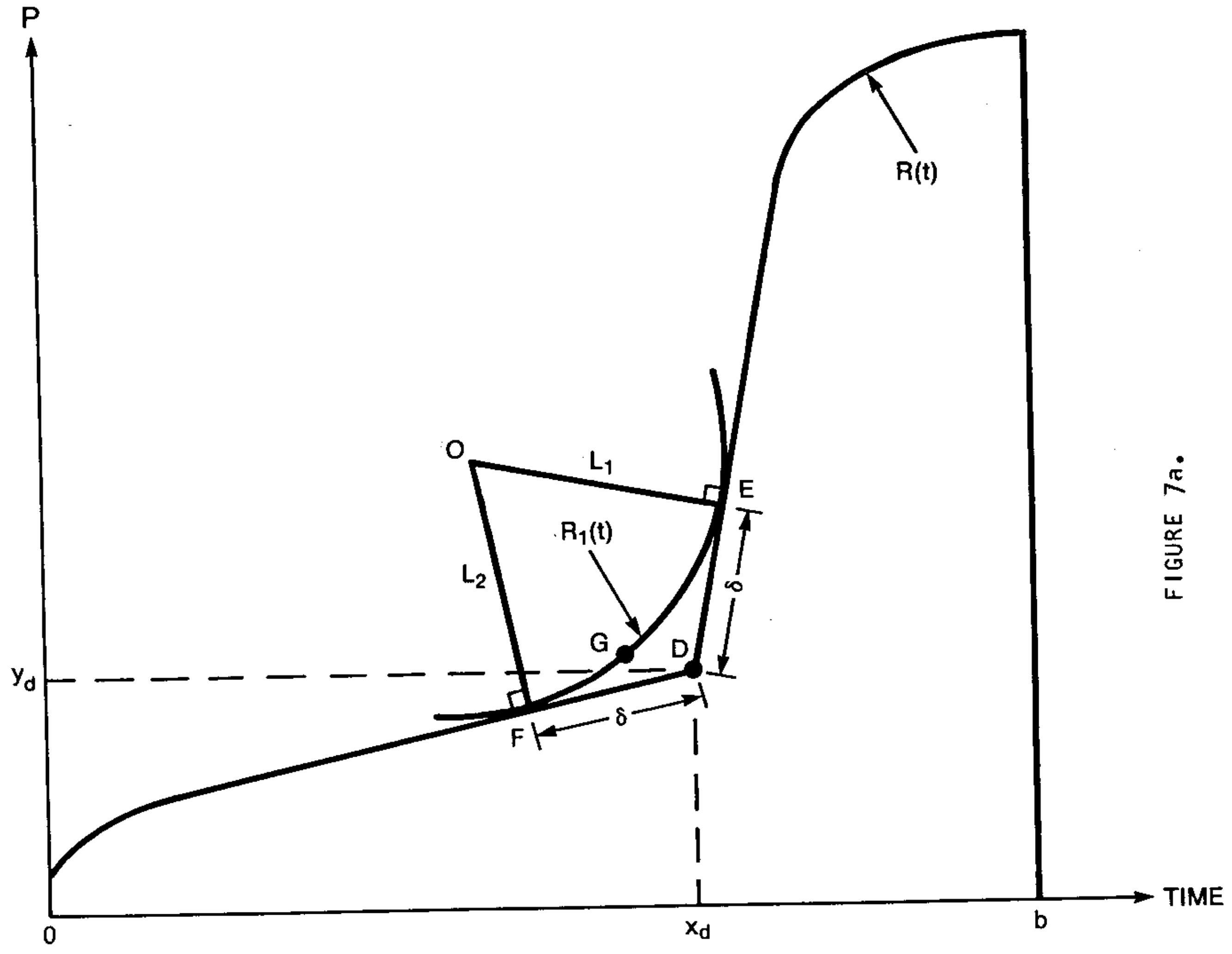


FIGURE 7a.

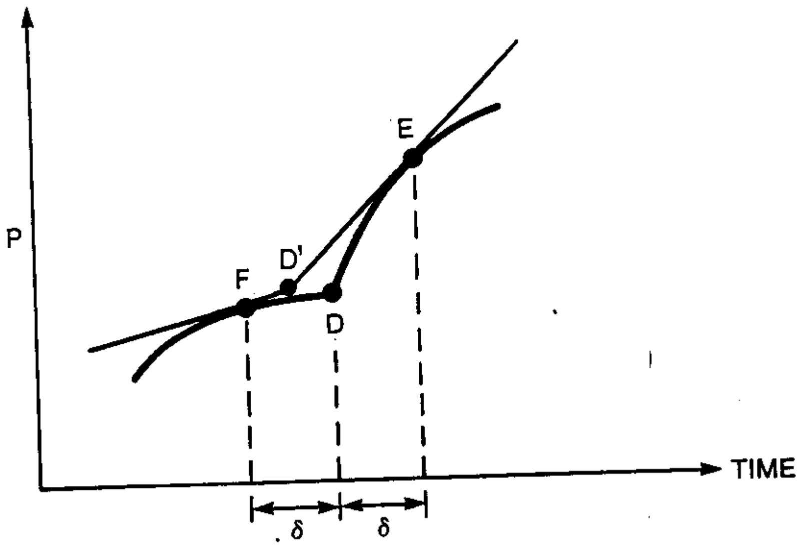


FIGURE 7b.

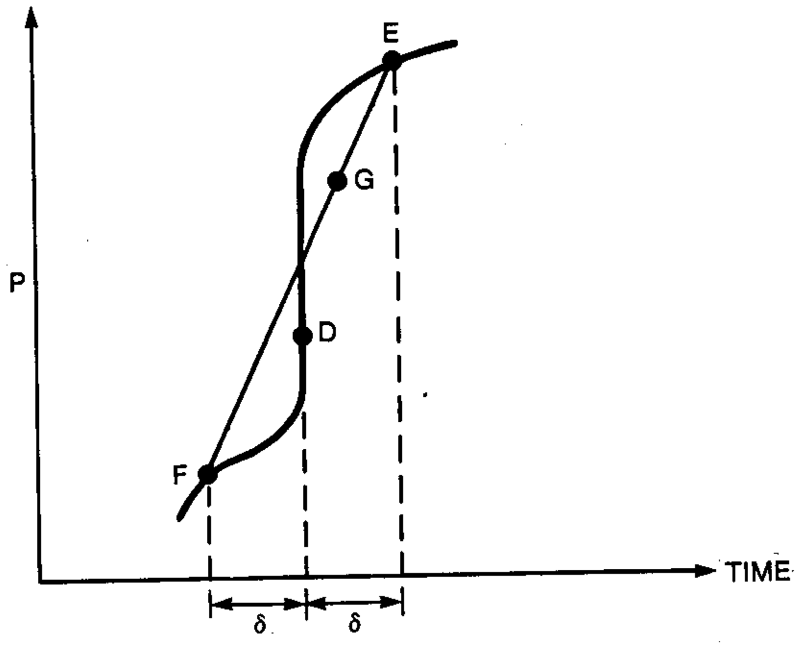


FIGURE 7c.

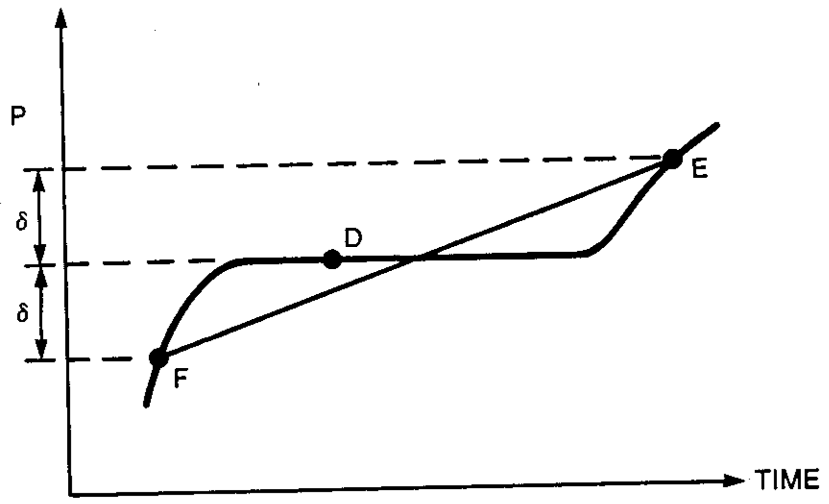


FIGURE 7d.