On Parallel Processing Systems: Amdahl's Law Generalized and Some Results on Optimal Design

Leonard Kleinrock, *Fellow, IEEE,* and Jau-Hsiung Huang

Abstract—**We model a job in a parallel processing system as a sequence 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 = throughput/response time) as previously applied to general queueing and computer-communication systems to our case of parallel processing systems. With this definition of power we are able to find the optimal system operating point (i.e., the optimal 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 is a continuous variable that changes continuously over time. The same performance indices and parameters studied in the discrete model are evaluated for this continuous model. These continuous results are more easily obtained, are easier to state, and are simpler to interpret than for the discrete model.

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

I. Introduction

As parallel computing systems proliferate, the need for effective performance evaluation techniques becomes ever more important. In this paper, we study certain fundamental performance indices, namely, *speedup*, *response time*, *efficiency*, and *power*, and solve for the optimal operating point of these systems. Specifically, by maximizing "power," we are able to find the optimal input rate of jobs and the optimal number of processors to use, given a characterization of the workload.

We model a parallel processing system as a system with a single queue of waiting jobs. Our first model (in Section IV) assumes that only a single job needs to be processed. Our second model (in Section V) allows a stream of arrivals to enter the system; however, only one job may be admitted into service at a time, following a FCFS discipline, while the others wait in the queue. Both models deal with jobs as follows. While in service, the system provides a maximum of P parallel processors to work on the job. A job is modeled as a sequence of independent stages which must be processed, where the number of processors desired by the job in each stage may be different. If, for some stage, the job in service requires fewer processors than the system provides, then the job will 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 system provides, then it will use all the processors in the system (in a processor sharing fashion [10]) for an extended period of time such that the total work served in that stage is conserved. An example is given in Fig. 1 in which the total processing work required by a job is $W = 24$ s. In this example, if $P \geq 6$, then it takes 8 s to complete the job as shown in Fig. 1(a), whereas if only $P = 4$ processors are provided, then it takes 9 s as shown in

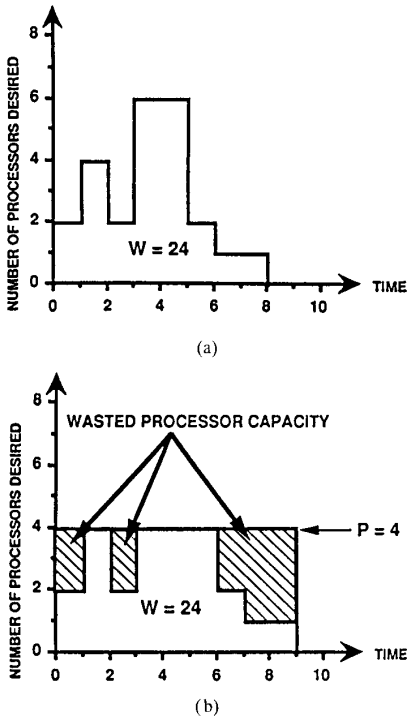


Fig. 1. Job profile. (a) Unlimited number of processors. (b) Limited number of processors ($P = 4$).

Manuscript received April 1, 1991; revised September 20, 1991. Recommended by E. Gelenbe. This work was supported by the Defense Advanced Research Projects Agency, Department of Defense under Contract MDA903-87-C-0663.

L. Kleinrock is with the Computer Science Department, University of California, Los Angeles, Los Angeles, CA 90024.

J.-H. Huang is with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan.

IEEE Log Number 9105395.

0098–5589/92$03.00 © 1992 IEEE

Fig. 1(b), in which case 12 s of processor capacity are wasted.

The model described above has been highly idealized. In particular, we are neglecting some of the following important aspects of the workload. First, we do not allow general precedence relations among the tasks. Our precedence structure is equivalent to a series-parallel task graph with deterministic task service times (see [6] for the definition of the task graph model of computation). Second, we do not separately model the communication times between tasks (i.e., the interprocess communication overhead). We hasten to point out that incorporating this overhead is not simply a matter of adding additional time to each task's processing time, since such overhead only occurs when a task on one processor must pass its results to a task on a different processor; thus to properly include interprocess communication costs, one must model the way in which tasks are assigned to processors (i.e., the task partitioning problem), an assignment that we choose to neglect. Third, we ignore I/O communication overhead related to the management and execution of parallel programs. Lastly, we assume that the program structure is infinitely divisible, in that the time to execute w units of work is equal to $\max\ (w/P, w/P')$, where P is the number of processors that the system provides for execution of this work, and P' is the maximum number of processors that the program is able to use for this work (i.e., the parallelism for this work). These assumptions simplify our analysis and lead to idealized results.

Our workload model was first reported by us in [8]. Later, Gelenbe [6] described a very similar model, as did Sevcik [15]. Gelenbe extended his model, which he referred to as the "Activity Set Model," to include the effect of inefficient use of processors, imbalance of the workload among the processors, and interprocess communication times. Sevcik also described ways in which this idealized model could be extended to include the effect of I/O communications, overhead, and dependencies among parallel threads assigned to different processors.

For such a parallel processing system there are two performance measures which compete with each other: *processor efficiency* and *mean response time.* One can increase the processor efficiency of the system (by reducing the number of processors), but then the mean response time will also be increased. Similarly, one can lower the mean response time (by increasing the number of processors), but then 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. We seek to find that number of processors which maximizes power.

Power, studied in [5], [11], and [12], was defined for a general queueing system in [12] as

$$\frac{\rho}{T/\overline{x}}$$

where ρ is defined as the system utilization, T is defined as the mean response time, and $\overline{x}$ 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 since $0 \leq \rho < 1$, and since $1 \leq T/\overline{x}$, then $0 \leq$ power < 1.) The symbol "*" will be used throughout to denote variables which are optimized with respect to power. In [12] it was found that for any M/G/1 queueing system [9], power is maximized when $\overline{N}^* = 1$, where $\overline{N}$ = 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 show that $\overline{N}^* = 1$ when power is maximized with respect to the job arrival rate (λ).

One might argue that power, as here defined, is an arbitrary performance measure. In response to this argument we point out that one can generalize the definition of power in a way which allows the reader to emphasize delay (or efficiency) in a variety of ways so as to match his or her needs. This issue is discussed below in Section II as well as in [5] and [12]. Moreover, other researchers have seen fit to optimize power for models similar to ours (see, for example [4]). An extensive study of power applied to computer networks is given in [5].

An alternative, and much more familiar, performance measure for parallel processing systems is *speedup*, which describes how much faster a job can be processed using multiple processors, as compared to using a single processor. Specifically, speedup is the ratio of the mean response time of a job processed by a single processor to that of a job executed in a parallel processing system with, say, P processors. Speedup and power are related and we discuss how they interact throughout this paper. Eager *et al.* [4] also discuss issues similar to those in this paper. Their focus is on estimating speedup and efficiency (for the no arrivals case only) simply from the value of the "average parallelism," which is defined as W, the total processing work required by a job, divided by the time it would take to service the job if there were an unlimited number of processors available; in Fig. 1(a) we have $W = 24$, and service time $= 8$, giving an average parallelism equal to 3. They also use the definition of power as we had defined in [11] and [12] and obtain the same result as we obtain in Corollary 7 below. They consider the case of deterministic workloads. Gelenbe [6] introduced an alternate model for the workload for which he also calculates speedup in the case of an infinite number of available processors. He models a job as having a random task graph in which the density of precedence relations between tasks is given by p $(0 \leq p \leq 1)$; he then derives an approximation for an upper bound on the speedup; namely, $(1 + p)/2p$.

II. Definitions

We have already defined the following:

$P =$ Number of (identical) processors in the server;
$W =$ Average number of seconds required to process a job on a single processor; and
$\overline{N} =$ Average number of jobs in the system.

Moreover, we now define the following additional quantities:

$\overline{x}(P) =$ Mean service time of a job in a P-processor system (note that the maximum mean service time is $\overline{x}(1) = W$ and that the minimum mean service time is $\overline{x}(\infty)$);

$T(\lambda, P) =$ Mean response time (queueing time plus service time) of a job in a queueing system with an input rate λ and P processors;

$\lambda =$ arrival rate of jobs;

$\rho =$ system utilization; i.e., the fraction of time when there is at least one job in the system.
$= \lambda \overline{x}(P)$; and

$u(P) =$ processor efficiency in a P-processor system.

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" in that case, since there may be some idle processors (i.e., it may be that 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.)

Two cases regarding the number of jobs in the system are considered in this paper. Case one allows no arrivals of additional jobs (Section IV). That is, there is only one job in the system, and we are concerned with $\overline{x}(P)$, its mean service time in a P-processor system. Case two allows jobs to arrive from a Poisson process at a rate λ, and so queueing effects are considered (Section V).

For the first case, we define the (no arrivals case) *speedup* with P processors, denoted by $S_n(P)$, to be

$$S_n(P) = \frac{\overline{x}(1)}{\overline{x}(P)} = \frac{W}{\overline{x}(P)}.$$

Note that

$$1 = \frac{W}{\overline{x}(1)} \leq S_n(P) \leq \frac{W}{\overline{x}(\infty)}.$$

Thus it is natural for us to define the maximum value for speedup $S_{n,\max}$ as follows:

$$S_{n,\max} = \frac{W}{\overline{x}(\infty)}.$$

Furthermore, we see that $S_{n,\max} =$ average parallelism.

For the second case, we define the (arrivals case) *speedup* with P processors at system utilization ρ, denoted by $S_a(\lambda, P)$, to be

$$S_a(\lambda, P) = \frac{T(\lambda, 1)}{T(\lambda, P)}.$$

We must distinguish the processor efficiency $u(P)$ in these two cases as follows:

$u_n(P) =$ processor efficiency given P processors in the no arrivals case; and

$u_a(\lambda, P) =$ processor efficiency given job arrival rate λ and P processors in the case with job arrivals.

We now introduce the appropriate definitions of *power*, which we denote by the symbol Q (we would prefer to use the obvious notation P, but P has already been used to denote the number of processors). Let

$Q_n(P) =$ power given P processors in the no arrivals case; and

$Q_a(\lambda, P) =$ power given a job arrival rate λ and P processors in the case with job arrivals.

In this paper we are concerned mostly with power which is defined as processor efficiency divided by the mean response time.

In the case of no arrivals, the mean response time of the (single) job is simply its mean service time $\overline{x}(P)$, and so:

$$Q_n(P) = \frac{u_n(P)}{\overline{x}(P)}.$$

Clearly, power will increase by either raising the processor efficiency or by lowering the mean service time. A more general definition of power (as originally introduced in [12]) is given as

$$Q_n^{(r)}(P) = \frac{[u_n(P)]^r}{\overline{x}(P)}$$

where r is a positive real number whose value may be selected by the system designer. With this generalization, a designer may express a stronger preference for an increase in the processor efficiency at the expense of an increase in the mean service time by simply increasing the value of the parameter r (and vice-versa). Note that $Q_n(P) = Q_n^{(1)}(P)$.

In the case of job arrivals, the definition of power becomes:

$$Q_a(\lambda, P) = \frac{u_a(\lambda, P)}{T(\lambda, P)}$$

and the generalization in this case is

$$Q_a^{(r)}(\lambda, P) = \frac{[u_a(\lambda, P)]^r}{T(\lambda, P)}$$

where again r is a positive real number to be used as a degree of freedom by the system designer. Note that $Q_a(\lambda, P) = Q_a^{(1)}(\lambda, P)$.

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

The rest of this paper is organized as follows. In Section III we present two models of a job: a discrete model, and a continuous model. In Section IV we solve the case when no arrivals are allowed in the system. In this case we find the speedup of the system given P processors. We also find P^*, the number of processors which maximizes power. In Section V we solve the case when job arrivals are allowed in the system. In this case we again solve for the 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 simplification in system design.

III. Workload Models

We consider both a discrete as well as a continuous model of job requirements.

A. A Discrete Job Model

Here, we model a job as containing a total of $\widetilde{W}$ tasks. Nonoverlapping subsets of these tasks are collected into *stages*, and these stages are processed sequentially (however, parallelism is exploited *within* each stage—see below). $\widetilde{W}$ is a random variable with mean W and coefficient of variation c_W[1]. We assume that the service time distribution for each task is deterministic, such that each task requires 1 s of work on a processor. For the results we seek in this paper, a job is described by specifying W and c_W along with two other vectors. The first vector is called the *fraction* vector, $\boldsymbol{f}'$, and the second vector is called the *processor* vector, $\boldsymbol{P}'$. We denote the fraction and processor vectors as

$$\boldsymbol{f}' = [f_1', f_2', f_3', \cdots, f_{n'}']$$
$$\boldsymbol{P}' = [P_1', P_2', P_3', \cdots, P_{n'}']$$

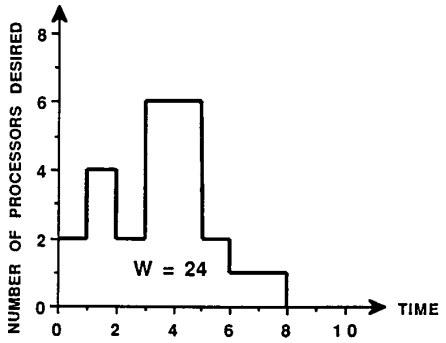
where n' is the number of stages in a job. The ith stage has the pair (f_i', P_i') associated with it. The meaning is as follows: a fraction f_i' of the total tasks in a job can use P_i' processors to concurrently process these tasks. For this definition, it is clear that

$$\sum_{i=1}^{n'} f_i' = 1.$$

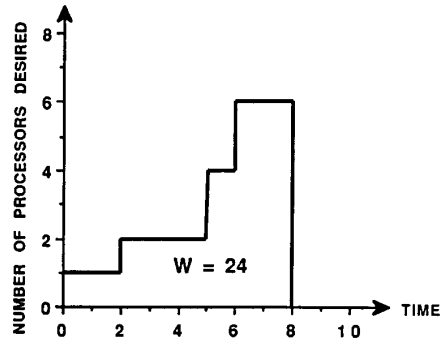
The example from Fig. 1 is repeated in Fig. 2(a), where $W = 24$ and $n' = 6$. Stage 4 contains 12 tasks, and so $f_4' = 1/2$; moreover, since $P_4' = 6$ (and if $P \geq 6$), then it will take 2 s to complete stage 4. This stage-type workload model comes directly from the usual task graph model of computation [3] with deterministic task service times. The ith stage corresponds to the ith level in the computation graph.

For convenience, we may rearrange the elements in $\boldsymbol{f}'$ and $\boldsymbol{P}'$ as follows in such a way that neither the mean response time nor the processor efficiency are changed. The elements of $\boldsymbol{P}'$ are rearranged and renumbered so that its elements are nondecreasing; that is, $P_{i-1}' \leq P_i'$. The elements of $\boldsymbol{f}'$ follow the identical permutation and renumbering. We may then merge several stages with the same P_i''s into one stage simply by adding all the corresponding f_i''s. The new vectors will be denoted $\boldsymbol{P} = [P_1, P_2, \cdots, P_n]$ and $\boldsymbol{f} = [f_1, f_2, \cdots, f_n]$, where $n \leq n'$ and $P_{i-1} < P_i$. 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. The example in Fig. 2(a) has been rearranged as shown in Fig. 2(b), where the number of stages is now $n = 4$. Note that $\overline{x}(\infty) = 8$, as it was in Fig. 2(a). One can easily see that if we choose $P = 4$, then $\overline{x}(4)$ will equal 9 in this rearranged case, as was the case for Fig. 1(b).

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



(a)



(b)

Fig. 2. Rearranging the job profile. (a) $\boldsymbol{P} = [2, 4, 2, 6, 2, 1]$, $\boldsymbol{f} = [\frac{1}{12}, \frac{1}{6}, \frac{1}{12}, \frac{1}{2}, \frac{1}{12}, \frac{1}{12}]$. (b) $\boldsymbol{P} = [1, 2, 4, 6]$, $\boldsymbol{f} = [\frac{1}{12}, \frac{1}{4}, \frac{1}{6}, \frac{1}{2}]$.

B. A Continuous Job Model

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 (not necessarily discrete) nondecreasing function of time (recall the rearranging does not affect performance). That is, we permit nonintegral numbers of processors (which could correspond to cases where processors are shared among more than one job). A special model with a deterministic workload per job will be described first, and then a more general model with a random workload per job will be described.

For the special case with a deterministic workload, we define $P(t) = g(t)$, where $g(t)$ is a deterministic function, to be the number of processors that a job desires at time t $(0 \leq t \leq b)$ such that $P(b) = B$ (see Fig. 3). For such a model, the workload (seconds of work required) for each job is deterministic with value

$$W = \int_0^b P(t)\, dt.$$

Note that $b = \overline{x}(\infty)$. Moreover, if we limit the number of processors to $P(P < B)$, then A, the (shaded) area of $P(t)$ which lies above the value of P, will be flattened out and extended as a rectangle of area A and of height P beginning

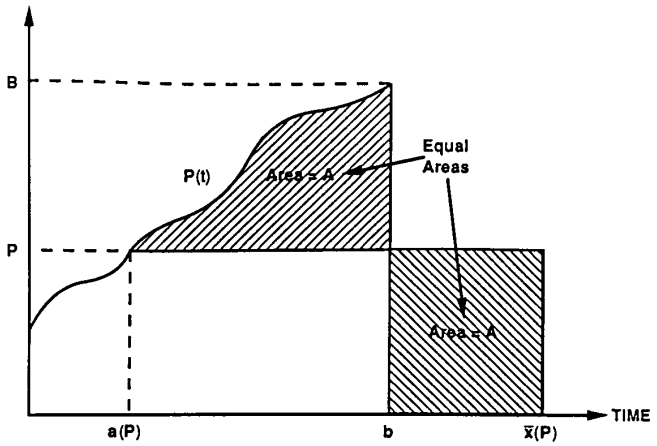


Fig. 3. A continuous job profile and the effect of a limited number of processors.

at the point b on the time axis and extending to the new mean service time $\overline{x}(P)$, as shown in Fig. 3.

For the general case with a random workload, we define

$$\widetilde{P}(t) = g\left(\frac{t}{\widetilde{K}}\right)$$

to be a random function which gives the number of processors desired by the job at time t. $\widetilde{K}$ 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, with $0 \le t \le \widetilde{K}b$. The distribution of $\widetilde{K}$ affects the results given in this paper in the following simple fashion:

i) All time variables should be multiplied by $\widetilde{K}$;
ii) All mean time variables should be multiplied by $\overline{K}$; and
iii) The optimal value of P, namely P^*, is independent of the distribution of $\widetilde{K}$.

Therefore we assume that $\overline{K} = 1$ in the remainder of this paper at no loss of generality. In addition, c_K does affect some of our later expressions, and in those cases it will appear explicitly.

We remind the reader of some of the limitations of this workload model as listed in the introduction. These limitations include: the lack of full generality in the precedence structure among the tasks; the neglect of the interprocess communication overhead; the neglect of I/O processing and communication overhead; and the assumption of the infinite divisibility of the workload.

IV. Systems With No Arrivals

In this section we examine a system of P processors serving a single job with no new arrivals. We wish to find the speedup and the value P^* which maximizes power for the processing of this job. For systems with no arrivals, we have that the processor efficiency is

$$u_n(P) = \frac{W}{P\overline{x}(P)}.$$

This follows from our earlier definition, since the system is busy for $\overline{x}(P)$ s, and in this time P processors could do $P\overline{x}(P)$ s of work; however, they only accomplish W s of work, since some processors may occasionally be idle. From the definitions of speedup $S_n(P)$ and power $Q_n(P)$, we see that only W, P, and $\overline{x}(P)$ are involved in the definitions. Since W, P, and $\overline{x}(P)$ are not affected by the distribution of the service requirement (but only by the mean W), we conclude that both the speedup and power only depend upon the mean values. Therefore we will not see any higher moments of the job service requirements in this section. Jobs with different random service requirements will give the same results as long as they have the same means. We first study the continuous case, and then apply the results obtained to the discrete case.

A. The Continuous Case

We consider a job profile $P(t)$ such as that shown in Fig. 3. We wish to determine P^*, the optimum number of processors to use with this job, where power is the objective function we seek to maximize. We define:

$a(P) =$ length of the interval from when the job first begins service until it first requires more processors than the system supplies; i.e., $a(P) = \min(t : P(t) > P)$;

$b =$ 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; i.e., $P \ge B$ (note that $b = \overline{x}(\infty)$); and

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

Note from Fig. 3 that $I(P) = [\overline{x}(P) - a(P)]P$. The average service time is

$$\overline{x}(P) = a(P) + \frac{\int_{a(P)}^{b} P(t)\,dt}{P}.$$

The speedup for this system is simply:

$$S_n(P) = \frac{W}{\overline{x}(P)} = \frac{PW}{Pa(P) + \int_{a(P)}^{b} P(t)\,dt}.$$

Theorem 1

The power $Q_n(P)$ is maximized with respect to P when $P = P^*$, where P^* is the unique (typically nonintegral) value satisfying

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

Proof: Since

$$u_n(P) = \frac{W}{P\overline{x}(P)}$$

and

$$Q_n(P) = \frac{u_n(P)}{\overline{x}(P)}$$

we have:

$$Q_n(P) = \frac{W}{P[\overline{x}(P)]^2}.$$

Maximizing $Q_n(P)$ with respect to P, we require:

$$\frac{d}{dP} Q_n(P) = 0$$

which leads to the general condition

$$\frac{d\overline{x}(P)}{dP} = -\frac{\overline{x}(P)}{2P}.$$

At this point one must consider any problems which might arise in calculating $(d\overline{x}(P))/(dP)$, if $P(t)$ has: (i) any nondifferentiable points, or (ii) any vertical jumps, or (iii) any horizontal segments. However, it can easily be shown for these three cases that an infinitesimal change in P can only make an infinitesimal change in $\overline{x}(P)$, since the change in the area A must be continuous (see Fig. 3). The only troublesome case is case (iii), since for any $\varepsilon > 0$, $a(P_0 + \varepsilon) - a(P_0 - \varepsilon) = c$ when $P(t)$ has a horizontal segment of length c and height P_0. This is troublesome, since $\overline{x}(P) = a(P) + (I(P))/P$, and thus $\overline{x}(P)$ appears to have a discontinuous jump at $P = P_0$; however, the term $(I(P))/P$ has an equal and opposite jump there as well, which eliminates the problem. Nevertheless, we will indeed run into a problem in uniquely defining $a(P)$ in Corollary 1, below, if $P^* = P_0$, where P_0 is any such horizontal segment height; we settle this problem further below in Corollary 7, case (ii).

From our expression for $\overline{x}(P)$ we find that the general condition given above leads us to the following expression which must be satisfied by the optimal value of P:

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

Now, since $I(P) = \int_{a(P)}^{b} P(t)\,dt$, we have

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

But $a(P)$ is such that $P(a(P)) = P$, and so $(dI(P))/(da(P)) = -P$. Using the chain rule, we then have $(dI(P))/(dP) = -P(da(P))/(dP)$; therefore:

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

Thus we see that the optimal value P^* must be such that

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

It can easily be shown that $(d^2Q_n(P)) < 0$; therefore $P^* = (I(P^*))/(a(P^*))$ indeed maximizes power. □

The expression for P^* as given in theorem 1 is not especially illuminating. To help explain the meaning of theorem 1, let us state and then interpret the following corollary:

Corollary 1

Power is maximized if and only if

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

Proof: Since $(I(P^*)/P^*) = a(P^*)$, and from the definition of $\overline{x}(P)$ we have that $\overline{x}(P^*) = a(P^*) + a(P^*) = 2a(P^*)$. □

Corollary 1 is one of the principal results of this paper. To interpret this corollary we note from Fig. 3 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 for a job 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 $(I(P))/P$ to be the "extended service time." This corollary states that the optimal number of processors P^* must be selected so that the "unextended service time" exactly 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 $u = 1$.

At this point we may simplify the proof of the following theorem which appeared in [4, theorem 5]:

Theorem

Under the processor sharing discipline when the number of available processors is equal to P^*, the attained speedup is *at least* 50% of the maximum possible, the efficiency is *at least* 50%, the utilization of the last processor is *at least* 50%, and the utilization of a single additional processor is *no more than* 50%. These bounds can be achieved in the limit as $S_{n,\max} \to (\infty)$.

Proof: For any P we know that $a(P) \leq b \leq \overline{x}(P)$. For $P = P^*$, we know that $\overline{x}(P) = 2a(P)$, and so $\overline{x}(P) \leq 2b = 2\overline{x}(\infty)$. Thus

$$S_n(P) = \frac{W}{\overline{x}(P)} \geq \frac{W}{2\overline{x}(\infty)} = \frac{S_{n,\max}}{2}.$$

That is, $S_n(P)$ is at least half of the maximum achievable speedup. Moreover, all processors are continuously busy in the interval $a(P) \leq t \leq 2a(P)$, and some others may be busy in the interval $0 \leq t \leq a(P)$; thus the processor efficiency is at least 50%. Clearly, the last processor added is busy half the time $(a(P) \leq t \leq 2a(P))$. Any additional processor beyond P^* will be busy only during the interval

$a(P+1) \leq t \leq \bar{x}(P+1)$; but $a(P) < a(P+1)$ and $\bar{x}(P) = 2a(P) > \bar{x}(P+1)$, and so this additional processor efficiency is less than 50%. The attainment of the bounds in the limit is obvious. □

Let us now state the results for the case in which we select $P = P^*$ to optimize the generalized power function $Q_n^{(r)}(P)$.

Theorem 2

Generalized power

$$Q_n^{(r)}(P) = \frac{[u_n(P)]^r}{\bar{x}(P)}$$

is maximized when P^* is selected, such that

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

Proof: This theorem can easily be derived following the procedure given in the proof for theorem 1. □

Corollary 2

$Q_n^{(r)}(P)$ is maximized when P^* is selected, such that

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

Proof: This corollary can easily be derived following the procedure given in the proof for corollary 1. □

From corollary 2 we easily generalize the theorem [4] discussed above in the form of the following:

Theorem 3

When P^* is selected to optimize $Q_n^{(r)}(P)$, then the attained speedup $S_n(P)$ is at least a fraction $1/(r+1)$ of the maximum possible, the processor efficiency is at least $r/(r+1)$, the efficiency of the last processor added is at least $r/(r+1)$, and the efficiency of a single additional processor is no more than $r/(r+1)$.

Proof: This theorem can easily be derived following the procedure given by us in the simplified proof of the theorem [4] above. □

It is instructive to graph the result of theorem 3 as in Fig. 4. In this figure we indicate that when $P = P^*$, then one is guaranteed to lie in the shaded region. As r varies, the shaded rectangle moves from a tall thin rectangle near the right-hand border ($r \to 0$) to a square occupying the upper right-hand quarter of the figure (at $r = 1$) to a wide flat rectangle along the top border of the figure ($r \to (\infty)$). The theorem from [4] basically states the case only for $r = 1$.

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

Example 1

If $P(t)$ is a linear function—i.e., $P(t) = (B/b)t$ for $0 \leq t \leq b$, and P^* is chosen to maximize the power function $Q_n^{(r)}(P)$, then we have

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

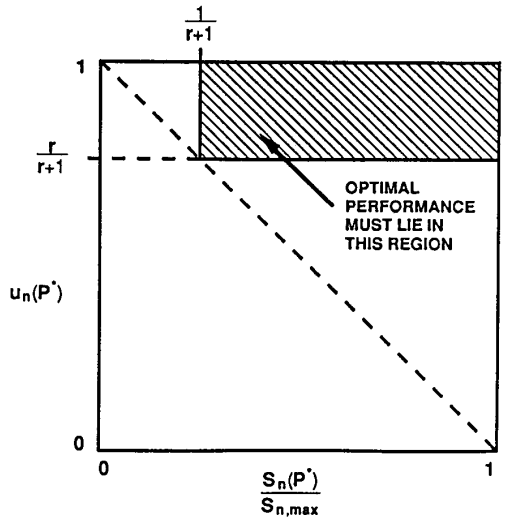


Fig. 4. Region of optimal performance.

or

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

Proof: From $P(t)$ we have

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

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

Hence, since $P^* = (I(P^*))/(ra(P^*))$, we have

$$P^* = \frac{B^2 - P^{*2}}{2rP^*}.$$

Solving for P^*, we have

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

Note that B is the maximum number of processors required by the job. □

In this example, by setting $r = 1$, we have $P^*/B = 1/\sqrt{3} \cong 58\%$. This is the case that was solved approximately in [13] by numerically solving a 5th-order polynomial; here we have found the *exact* value of P^* analytically.

Example 2

If $P(t) = B/(b^n)t^n$ for $0 \leq t \leq b$, and P^* maximizes $Q_n^{(r)}(P)$, then

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

or

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

Proof: This proof is similar to the proof in example 1. □

We are now in a position to apply some of these results to the simple case of a job with two discrete stages.

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 $(1 - f)$ of the tasks can be done concurrently with at most P processors, where P is the number of processors in the system. Note that this is the model used by Amdahl to derive Amdahl's Law [1], a classic result which we now state.

Amdahl's Law

The speedup of this model, given P processors, is upper bounded as follows:

$$S_n(P) \leq \frac{P}{fP + 1 - f}.$$

Proof: Since Wf of the tasks must be done serially, they will take Wf s; also, since $(1 - f)W$ of the tasks can be done concurrently with at most P processors, they will take at least $((1 - f)W)/P$ s. Therefore the mean service time $\bar{x}(P)$ is at least $Wf + ((1 - f)W)/P$. Moreover, since a single processor working alone will take W s, the speedup is simply W divided by $\bar{x}(P)$, which proves the result. □

This law implies that the speedup of the system depends very strongly on the simple workload measure f, and that the speedup may be much smaller than the number of processors, even for a relatively small f. For example, if $f = 0.1$, then one will, at best, obtain a speedup of less than 10 with 1000 times the processing capacity $(P = 1000)$[2].

For the remainder of this subsection we will make the (optimistic) assumption that $1 - f$ of the tasks can use *exactly* P processors concurrently. For this revised model, the upper bound in Amdahl's Law will be achieved. This corresponds to our discrete model of jobs for which $\boldsymbol{f} = [f, 1 - f]$, and $\boldsymbol{P} = [1, P]$. For a given value of f, we solve for the optimum value of P in the next theorem.

Theorem 4

Power $Q_n(P)$ is maximized when the number of processors is selected to be[3]

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

Proof: During the entire service time $\bar{x}(P) = Wf + (W(1 - f))/P$, 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:

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

Thus

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

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

The result given in theorem 4 is intuitively pleasing. Fig. 5 shows the curve for P^* versus f. Note the sharp drop in P^* when f is small, and also note that $P^* = 1$ for $f \geq 1/2$.

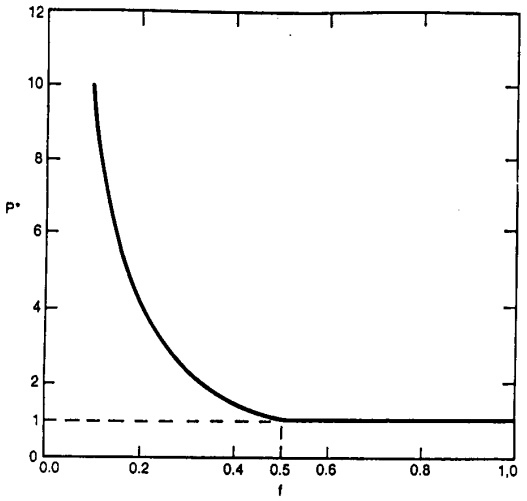


Fig. 5. The optimal number of processors for two stages ($\boldsymbol{f} = [f, 1 - f]$ and $\boldsymbol{P} = [1, P]$).

Corollary 3

For $P^* = (1 - f)/f > 1$, the interval of time when the system is working on the serial portion of the job exactly equals the interval of time when the system is working on the parallel portion.

Proof: The service time for the serial portion of fW. The service time for the parallel portion is

$$\frac{(1 - f)W}{P^*} = fW. \qquad \square$$

[2] As a result of Amdahl's law, one is easily discouraged from using parallel processing. Nevertheless, experience shows in a number of cases that speedups very close to P are quite possible [2]. Gustafson explains this [7] by suggesting that as the number of processors increases, the application problem size also increases in a way such that the parallel portion of the problem grows while the serial portion remains fixed; that is, $f = f(P)$ is a decreasing function of P. In this paper we assume that f is constant, independent of P. Gelenbe [6] provides analytical evidence of this linear growth of the speedup with $\boldsymbol{P}$ by considering a model which includes the effect of a program's inability to effectively use all of the processors assigned to it, as well as the effect of imbalance of the workload across the available processors; Gelenbe shows that this linear dependence on $\boldsymbol{P}$ may be lost, however, when the effect of interprocess communication is included in the model.

[3] In this case we require $P^* \geq 1$.

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, the speedup is as follows:

$$S_n(P^*) = \begin{cases} 1, & \text{if } f \geq \frac{1}{2} \\ \frac{1}{2f}, & \text{if } f < \frac{1}{2}. \end{cases}$$

Proof: From Amdahl's Law and our optimistic assumption, we have

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

From theorem 4 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 $S_n(P)$ completes the proof. □

Theorem 5

Generalized power $Q_n^{(r)}(P)$ is maximized when P^* is selected such that

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

Proof: It can be shown that

$$Q_n(P) = \frac{[u_n(P)]^r}{\overline{x}(P)} = \frac{1}{W} \cdot \frac{P}{(Pf + 1 - f)^{r+1}}.$$

Optimizing $Q_n^{(r)}(P)$ with respect to P, one finds that $P^* = (1 - f)/(rf)$. However, P^* must not be smaller than 1; hence $P^* = 1$, if $(1 - f)/(rf) \leq 1$ (or $f \geq 1/(r + 1)$). □

Corollary 5

For $P^* = (1 - f)/(rf) > 1$, the parallel portion of the job takes exactly r times as long to serve as does the serial portion of the job.

Proof: The service time for the serial portion is fW. The service time for the parallel portion is

$$\frac{(1 - f)W}{P^*} = r \cdot fW.$$ □

Corollary 6

At the optimal power point,

$$S_n(P^*) = \begin{cases} 1, & \text{if } f \geq \frac{1}{2} \\ \frac{1}{(r+1)f}, & \text{if } f < \frac{1}{2}. \end{cases}$$

Proof: This proof is similar to the proof for corollary 4. □

C. The Discrete Case in General

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

$$\boldsymbol{f} = [f_1, f_2, f_3, \cdots, f_n]$$
$$\boldsymbol{P} = [P_1, P_2, P_3, \cdots, P_n]$$

where $P_i < P_{i+1}$ for $1 < i < n - 1$, and

$$\sum_{i=1}^{n} f_i = 1.$$

Before describing the next theorem we need some more notation. We assume that there are P processors available in the system. We define the index "m" such that; if $P_1 \leq P < P_n$, then m is the integer that satisfies $P_{m-1} \leq P < P_m$; or if $P \geq P_n$, then $m = n + 1$ of if $P < P_1$, then $m = 1$. Once m determined, we may define:

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

and

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

Note that αW is the unextended service time, whereas $(\beta W)/P$ is the extended service time.

Theorem 6

The speedup for any P is given by

$$S_n(P) = \frac{P}{\alpha P + \beta}.$$

Proof: The mean service time is

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

Since a single processor working alone will take W s to serve a job, the speedup with P processors is simply W divided by $\overline{x}(P)$. □

We can easily modify this result to obtain a generalization of Amadahl's Law. Consider a job consisting of W tasks, for which a fraction f_i of these tasks can be done using *at most* P_i processors in parallel ($i = 1, 2, \cdots, n$). Then we have the theorem below.

Theorem 7 (Generalized Amdahl's Law)

For the system just described, the speedup, given P processors, is upper bounded as follows:

$$S_n(P) \leq \frac{P}{\alpha P + \beta}.$$

Proof: This result can easily be derived following the procedure to prove Amdahl's Law and theorem 6. □

The maximum possible speedup $S_{n,\max}$ for the model used in theorem 6 will be achieved when $P \geq P_n$, which gives

$$\alpha = \sum_{i=1}^{n} \frac{f_i}{P_i}$$

and $\beta = 0$. Hence

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

In the following two corollaries we derive P^* for the discrete case using the results from corollary 1. Corollary 7 was first obtained in [4, theorem 4].

Corollary 7[4]

Power $Q_n(P)$ is maximized when P^* satisfies either one of the following two conditions:

(i) $P^* = \dfrac{\beta}{\alpha}$, if $P_{m-1} < P^* < P_m$

or

(ii) $P^* = P_{m-1}$, if $0 \leq \dfrac{\alpha}{2} - \dfrac{\beta}{2P_{m-1}} \leq \dfrac{f_{m-1}}{P_{m-1}}$.

Proof: In case (i) there is no ambiguity in defining the unextended service time. Specifically, the unextended service time $= \alpha W$, and the extended service time $= (\beta W)/P$. From corollary 1 we must have

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

Hence

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

In case (ii) we 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 break the mean service time for stage $m-1$, namely, $t_{m-1} = f_{m-1}W/P_{m-1}$, into two segments: x and $t_{m-1} - x$. We define x to be the interval in stage $m-1$ which we include in the extended service time, and the interval $t_{m-1} - x$ to be the interval in stage $m-1$ which we conclude in the unextended service time, as shown in Fig. 6. To find x, we note from corollary 1 (and assuming $P^* = P_{m-1}$) that

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

which gives:

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

[4]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 be less than when $P^* = P_n$; hence the power will be smaller. A similar argument shows that P^* will never be smaller than P_1.

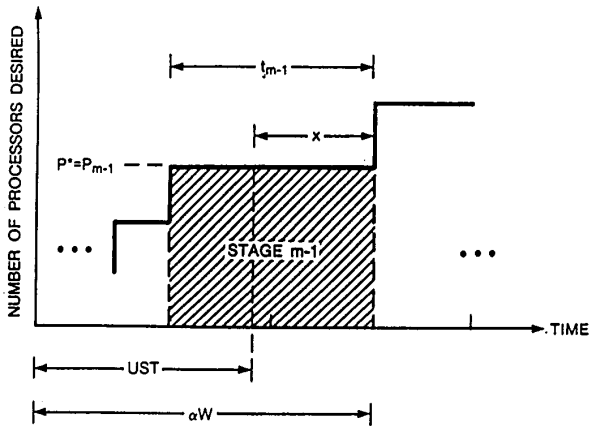


Fig. 6. Segmentation of the service time of stage $m-1$ (UST = Unextended Service Time).

Thus the unextended service time $a(P)$ is simply:

$$\begin{aligned} a(P) &= \alpha W - x \\ &= \frac{\alpha W + \frac{\beta W}{P_{m-1}}}{2} \\ &= \frac{\overline{x}(P)}{2} \end{aligned}$$

as demanded by corollary 1. Case (ii) will occur whenever, for some m, we have the following two conditions simultaneously true:

$$\alpha W - t_{m-1} < \frac{\beta W}{P_{m-1}} + t_{m-1}$$

$$\alpha W > \frac{\beta W}{P_{m-1}}.$$

These may be rewritten as

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

But since

$$t_{m-1} = \frac{f_{m-1}W}{P_{m-1}}$$

we have as the condition for case (ii):

$$0 \leq \frac{\alpha}{2} - \frac{\beta}{2P_{m-1}} \leq \frac{f_{m-1}}{P_{m-1}}.$$ □

Corollary 8

Generalized power $Q_n^{(r)}(P)$ is maximized when P^* satisfies either one of the following two conditions:

(i) $P^* = \dfrac{\beta}{r\alpha}$, if $P_{m-1} < P^* < P_m$

or

(ii) $P^* = P_{m-1}$, if $0 \leq \dfrac{r\alpha}{r+1} - \dfrac{\beta}{(r+1)P_{m-1}} \leq \dfrac{f_{m-1}}{P_{m-1}}$.

Proof: This proof is similar to that for corollary 7. □

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

In our results of Sections IV-B and -C we have neglected the physical requirement that P^* be an integer. Clearly, if P^* must be an integer, then P^* should then be rounded up or down to the nearest integer, whichever of the two has a larger value of power.

V. Systems With Arrivals

In this section we study the case when new jobs enter the system according to a Poisson process at rate λ. We now permit random service times. This model corresponds to a parallel processing system which executes one job at a time, but which can accept and enqueue new arrivals which are later served in a first-come-first-served fashion (one at a time). The following theorem describes a property which is useful in finding many 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 $P \geq 1$,

$$c_{x_1} = c_{x_P}.$$

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

$$\tilde{x}(P) = \tilde{K}\left[a(P) + \int_{a(P)}^{b} \frac{g(t)}{P}\,dt\right].$$

This equation 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

$$c_{x_P} = c_K.$$

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

For the two-stage discrete case we have

$$\tilde{x}(P) = \widetilde{W}f + \frac{(1-f)\widetilde{W}}{P} = \widetilde{W}\left[f + \frac{(1-f)}{P}\right]$$

where $\widetilde{W}$ is a random variable representing the work brought in by a job. Hence, using a similar argument as above, we have

$$c_{x_P} = c_{x_1} = c_W.$$

Similarly, for the general discrete case we have

$$\tilde{x}(P) = \widetilde{W}\alpha + \frac{\beta\widetilde{W}}{P} = \widetilde{W}\left[\alpha + \frac{\beta}{P}\right].$$

Hence, using the same argument as above, we have

$$c_{x_P} = c_{x_1} = c_W.$$ □

We can show that

$$\widetilde{W} = \int_0^{\tilde{K}b} g\left(\frac{t}{\tilde{K}}\right)dt = \tilde{K}\int_0^b g(t)dt.$$

Since

$$\int_0^b g(t)dt$$

is a constant, this equation shows that the work brought in by a job is a random variable which has the same coefficient of variation as $\tilde{K}$.

A. Finding the Speedup

In this section we find the speedup for all cases. We discover that the speedup when queueing is allowed is the same as the speedup when queueing is not allowed!

Theorem 9

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

$$S(\lambda, P) = S_n(P).$$

Proof: We have defined ρ to be the system utilization; hence

$$\rho = \lambda\overline{x}(P).$$

Since only one job can be admitted into service at a time, this system can be analyzed as a single-server queueing system. Hence we can apply results from M/G/1 theory [8] to find the average response time for this system. That is,

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

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

$$S_a(\lambda, P) = \frac{T(\lambda, 1)}{T(\lambda, P)} = \frac{\overline{x}(1)}{\overline{x}(P)} = S_n(P).$$ □

Therefore the speedup $S_a(\lambda, P)$ and the speedup $S_n(P)$ are solely determined by the job specification $P(t)$ and P (and not affected by the system's operating point λ) in our models. (Another interesting model studied in [8] has $S_a(\lambda, P) \neq S_n(P)$).

Corollary 9

For the continuous model, we have

$$S_a(\lambda, P) = \frac{P\int_0^b P(t)\,dt}{Pa(P) + \int_{a(P)}^b P(t)\,dt}.$$

Proof: This can easily be proved from the expression for $\overline{x}(P)$ and theorem 9. □

Corollary 10

For the two-stage discrete model, we have

$$S_a(\lambda, P) = \frac{P}{fP + 1 - f}.$$

Proof: This can easily be proved from the optimistic model of Amdahl's Law and theorem 9. □

Corollary 11

For the general discrete model, we have

$$S_a(\lambda, P) = \frac{P}{\alpha P + \beta}.$$

Proof: This can easily be proved from theorem 6 and 9. □

B. The Optimal Arrival Rate

In this section we find the optimal operating point (λ^*) for both the discrete case and continuous case. Even though the definitions of power in this paper and in [12] are different (since $\rho \neq u_a(\lambda, P)$), the results obtained in both papers are the same. Therefore all the deterministic reasoning given in [12] also applies in this paper.

Theorem 10

Power $Q_a(\lambda, P)$ is maximized when $\lambda = \lambda^*$ (for a given P), such that

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

Proof: When we allow arrivals we must calculate the processor efficiency over all time. The rate at which seconds of work enter the system is λW, and the maximum rate at which the processors can discharge work is P. Thus

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

From M/G/1 theory we have

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

where $\rho = \lambda\overline{x}(P)$. Defining power as earlier, we have

$$Q_a(\lambda, P) = \frac{u_a(\lambda, P)}{T(\lambda, P)} = \frac{\lambda W}{P} \cdot \frac{2(1 - \lambda\overline{x}(P))}{2 + (c_{x_P}^2 - 1)\lambda\overline{x}(P)} \cdot \frac{1}{\overline{x}(P)}.$$

Maximizing power with respect to λ, we have

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

Corollary 12

When power is maximized with respect to λ,

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

and

$$\overline{N}^* = 1.$$

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 [14], it is easy to show that

$$\overline{N}^* = \lambda^* T(\lambda^*, P) = 1.$$ □

The result given above for $\overline{N}^*$ is intriguing. Indeed, $\overline{N}^* = 1$ corresponds to the same deterministic reasoning given in [12] and which is described in our introduction.

Theorem 11

Generalized power $Q_a^{(r)}(\lambda, P)$ is maximized when $\lambda = \lambda^*$ (for a given P) such that

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

where

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

Proof: This proof is similar to the proof for Theorem 10. □

Corollary 13

When power is maximized with respect to λ, then

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

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

If $r \gg 1$, we have

$$\lim_{r \gg 1} \rho^* = 1 - \frac{1}{r}$$

and

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

Note that the results in Theorem 11 and corollary 13 are the same as in [12].

C. The Optimal Number of Processors (P^*)

In this section we first study the relationship between $Q_n(P)$ and $Q_a(\lambda, P)$. 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 utilization $u_a(\lambda, P)$ for systems with arrivals in terms of the utilization $u_n(P)$ for system with no arrivals as follows:

$$\begin{aligned} u_a(\lambda, P) &= (\text{processor utilization}) \\ &= (\text{processor utilization}|\text{system busy}) \\ &\quad \cdot P[\text{system busy}] \\ &\quad + (\text{processor utilization}|\text{system idle}) \\ &\quad \cdot P[\text{system idle}] \\ &= (\text{processor utilization}|\text{system busy}) \\ &\quad \cdot P[\text{system busy}]. \end{aligned}$$

Thus we come to the simple conclusion that

$$u_a(\lambda, P) = u_n(P) \cdot \rho.$$

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

$$\begin{aligned} Q_a(\lambda, P) &= \frac{u_a(\lambda, P)}{T(\lambda, P)} = \frac{\rho u_n(P)}{T(\lambda, P)} \\ &= \frac{\rho}{T(\lambda, P)/\overline{x}(P)} \cdot \frac{u_n(P)}{\overline{x}(P)}. \end{aligned}$$

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

$$Q_a(\lambda, P) = \frac{2\rho(1-\rho)}{2 - \rho + \rho c_{x_P}^2} \cdot Q_n(P).$$

Note that $\rho/[T(\lambda, P)/\overline{x}(P)]$ is simply the normalized power discussed in [12] and in the introduction.

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

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

Note that

$$\frac{2\rho^*(1-\rho^*)}{2 - \rho^* + \rho^* c_{x_P}^2}$$

is only a function of c_{x_P} (since ρ^* is also a function of c_{x_P} only as shown in corollary 12) 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 $Q_a(\lambda^*, P)$ is the same as P^* for $Q_n(P)$. That is, for c_{x_P} not a function of P, we have systems with

$$P^*(\text{for systems with no arrivals}) = P^*(\text{for systems with arrivals}).$$

For the generalized definition of power, we have

$$\begin{aligned} Q_a^{(r)}(\lambda, P) &= \frac{[u_a(\lambda, P)]^r}{T(\lambda, P)} = \frac{\rho^r [u_n(P)]^r}{T(\lambda, P)} \\ &= \frac{\rho^r}{T(\lambda, P)/\overline{x}(P)} \cdot \frac{[u_n(P)]^r}{\overline{x}(P)} \\ &= \frac{2\rho^r(1-\rho)}{2 - \rho + \rho c_{x_P}^2} \cdot Q_n^{(r)}(P). \end{aligned}$$

Using the same argument as above (i.e., for c_{x_P} not a function of P), we have systems with the property

$$P^*(\text{for systems with no arrivals}) = P^*(\text{for systems with arrivals}).$$

Therefore all the results for evaluating P^* obtained in Section IV can be used here. However, not every model has this characteristic. In [8], another model is discussed in which c_{x_P} is indeed a function of P. In that case, a numerical procedure is required to find P^*.

Corollary 14

For the continuous model, power is maximized when P^* is chosen such that

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

and

$$\lambda^* = \frac{1}{a(P^*)\left(2 + \sqrt{2 + c_K^2}\right)}.$$

Proof: This is easily derived from theorems 1 and 10. □

Corollary 15

For the two-stage discrete model, power $Q_a(\lambda, P)$ is maximized when

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

and the optimal number of processors is

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

Proof: This is easily derived from theorems 4 and 10. □

VI. Conclusion

For the model which allows no arrivals we found the speedup ($S_n(P)$) for any P and for the optimal number of processors (P^*) which maximizes power. This $S_n(P)$ was shown to be a generalization of Amdahl's Law. For the model which allows arrivals we found the speedup ($S_a(\lambda, P)$) for any P, the optimal arrival rate (λ^*), and the optimal number of processors (P^*) which maximizes power. It was interesting to find that $S_n(P)$ is the same as $S_a(\lambda, P)$ for the models studied in this paper. It was also interesting 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!

Our results apply to an idealized workload model which neglects the degradation to system performance due to certain sources of overhead; consequently, these results must be viewed simply as approximate indicators of choices in any practical system design process.

References

[1] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," *Proc. AFIPS,* vol. 30, 1967.

[2] R. E. Benn, J. L. Gustafson, and R. E. Montry, "Development and analysis of scientific application programs on a 1024-processor hypercube," Sandia Nat. Labs., Albuquerque, NM, Tech. Rep. SAND 88-0317, Feb. 1988.

[3] E. G. Coffman and P. J. Denning, *Operating System Theory.* Englewood Cliffs, NJ: Prentice-Hall, 1973.

[4] D. L. Eager, J. Zahorjan, and E. D. Lazowska, "Speedup versus efficiency in parallel systems," *IEEE Trans. Computers,* vol. 38, pp. 408–423, Mar. 1989.

[5] H. R. Gail, "On the optimization of computer network power," Ph.D. diss., Computer Sci. Dept., UCLA, Sept. 1983.

[6] E. Gelenbe, *Multiprocessor Performance.* New York: Wiley, 1989.

[7] J. L. Gustafson, "Re-evaluating Amdahl's Law," *Commun. ACM,* vol. 31, no. 5, pp. 532–533, May 1988.

[8] J. Huang, "On the behavior of algorithms in a multiprocessing environment," Ph.D. diss., Computer Sci. Dept., UCLA, 1988.

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

[10] L. Kleinrock, *Queueing Systems,* vol. 2, *Computer Applications.* New York: Wiley-Interscience, 1976.

[11] L. Kleinrock, "On flow control in computer networks," in *Conf. Rec., Int. Conf. on Communications,* June 1978, vol. 2, pp. 27.2.1–27.2.5.

[12] L. Kleinrock, "Power and deterministic rules of thumb for probabilistic problems in computer communications," in *Conf. Rec., Int. Conf. on Communications,* June 1979, pp. 43.1.1–43.1.10.

[13] K. C. Kung, "Concurrency in parallel processing systems," Ph.D. diss., Computer Sci. Dept., UCLA, 1984.

[14] J. D. C. Little, "A proof of the queueing formula $L = \lambda W$," *Operations Res.,* vol. 9, pp. 383–387, 1961.

[15] K. C. Sevcik, "Characterizations of parallelism in applications and their use in scheduling," *Perform. Eval. Rev.,* vol. 17, no. 1, pp. 171–180, May 1989.

Leonard Kleinrock (S'55–M'64–SM'71–F'73) received the B.S. degree in electrical engineering from the City College of New York in 1957 (evening session), and the M.S.E.E. and Ph.D.E.E. degrees from the Massachusetts Institute of Technology in 1959 and 1963, respectively.

Since 1963 he has been on the faculty of the Computer Science Department at the University of California, Los Angeles, and he is currently Chair and Professor of that department. His research interests focus on performance evaluation of high-speed networks and parallel and distributed systems. He has had over 160 papers published and is the author of five books. He is the Principal Investigator for the DARPA Parallel Systems Laboratory contract at UCLA. He is also founder of the CEO of Technology Transfer Institute, a computer-communications seminar and consulting organization located in Santa Monica, CA.

Dr. Kleinrock is a member of the National Academy of Engineering, is a Guggenheim Fellow, and a member of the Computer Science and Technology Board of the National Research Council. He has received numerous best paper and teaching awards, including the ICC 1978 Prize-Winning Paper Award, the 1976 Lanchester Prize for outstanding work in Operations Research, and the Communications Society 1975 Leonard G. Abraham Prize Paper Award. In 1982, as well as having been selected to receive the C.C.N.Y. Townsend Harris Medal, he was co-winner of the L. M. Ericsson Prize presented by His Majesty King Carl Gustaf of Sweden for his outstanding contribution in packet switching technology. In July 1986 he received the 12th Marconi International Fellowship Award presented by His Royal Highness Prince Albert, brother of King Baudoin of Belgium, for his pioneering work in the field of computer networks. In the same year he received the UCLA Outstanding Teacher Award. In 1990 he received the ACM SIGCOMM award recognizing his seminal role in developing methods for analyzing packet network technology.

Jau-Hsiung Huang received the B.S. degree in electrical engineering from the National Taiwan University in 1981, and the M.S. and Ph.D. degrees in computer science from the University of California, Los Angeles in 1985 and 1988, respectively.

He joined the faculty at the National Taiwan University in 1988, where he is an Associate Professor in the Department of Computer Science and Information Engineering. His research interests include design and performance evaluation of high-speed networks, multimedia systems, and parallel and distributed systems.