Response Time Bounds for Parallel Processor Allocation Policies

Rajeev Agrawal
Rajesh K. Mansharamani
Mary K. Vernon

Technical Report #1152

June 1993

Response Time Bounds for Parallel Processor Allocation Policies †

RAJEEV AGRAWAL, RAJESH K. MANSHARAMANI, AND MARY K. VERNON‡

University of Wisconsin, Madison, Wisconsin

June 7, 1993

ABSTRACT. The first result of this paper is a lower bound on mean response time, under a very general workload model, per class of multiprogrammed parallel processor allocation policies. This bound is derived from the mean response time of the optimal uniprocessor scheduling policy that uses the same workload information as the class of parallel processor allocation policies. The derivation of the bound also suggests how tighter bounds can be obtained on a per policy basis in some cases. Key features of the workload model include general job demands, *available* parallelisms, execution rates, and inter-arrival times, with arbitrary dependencies among these workload variables. The second result is that for linear execution rates (i.e., perfect speedups) and for i.i.d. exponential job demands that are independent of everything else, the Preemptive Smallest Available Parallelism First policy is optimal among policies that use no explicit information about job demand. Likewise, among all *processor conserving* policies the Preemptive Largest Available Parallelism First policy is pessimal. For the same assumptions it is also shown that the performance of a processor conserving policy is best when every job can make use of all processors and is worst when all jobs are fully sequential. This third result leads to easily computable bounds on mean response time. The second and third results are shown to be sensitive to the assumption of exponential job demands. Finally, some quantitative results are given that illustrate the use and tightness of the derived bounds.

Categories and Subject Descriptors: C.1.2 [**Processor Architectures**]: Multiprocessors - *Parallel Processors*; C.4 [**Computer Systems Organization**]: Performance of Systems - *Modeling Techniques*; D.4.1 [**Operating Systems**]: Process Management - *Multiprocessing/Multiprogramming, Scheduling*; D.4.8 [**Operating Systems**]: Performance - *Queueing Theory, Stochastic Analysis*

General Terms: Performance

Additional Keywords and Phrases: Parallel System Performance, Uniprocessor Performance, Response Time Bounds, Optimal Scheduling Policies, Sample Path Analysis

1 Introduction

The problem of finding processor allocation policies that minimize the mean response time of parallel jobs on a general purpose parallel processor system has remained unsolvable to date. For example, determining the optimal schedule to minimize the mean response time of a set of parallel jobs with arbitrary task graphs and known task processing times is an NP-complete problem [16]. As a result, it is desirable to obtain performance bounds against which one can compare the performance of a given processor allocation policy. For example, a lower bound on mean response time for a class of policies is a good reference point against

†This research was partially supported by the National Science Foundation under grants CCR-9024144, CDA-9024618, and ECS-8919818.

‡Authors' Addresses: R. Agrawal. Department of Electrical and Computer Engineering, 1415 Johson Drive, Madison, WI 53706; email: agrawal@bandits.ece.wisc.edu. R. Mansharamani and M. Vernon. Department of Computer Sciences, 1210 W. Dayton St., Madison, WI 53706; email: {mansha, vernon}@cs.wisc.edu.

which one can compare the performance of a given policy in that class. A lower bound for a class of policies may not be tight for all policies under all workloads. If we make specific assumptions about the workload it may be possible to obtain tighter bounds. For example, a previous study [9] compares the performance of several policies against the optimal policy under a restricted set of workload assumptions.

The main result of this paper is a lower bound result on the mean response time per class of parallel processor allocation policies under very general workload assumptions, where class is defined by the information structure of the policy. The bound is obtained from the mean response time of the optimal uniprocessor scheduling policy that (1) has the same workload information as the class of parallel processor policies and (2) schedules the workload on a uniprocessor of power equal to the total processing power of the parallel processor system. The result holds for processor allocation policies that make use of explicit job demand information as well as those that do not. The lower bound result is derived for a very general workload model that includes general distribution of job demand, general distribution of available job parallelism, general job execution rates (that model communication and synchronization overheads as well as load imbalance), and general inter-arrival times, with arbitrary dependencies among these workload variables[1]. The derivation of the bound also suggests how tighter bounds can be obtained on a per-policy basis in some cases.

The next set of results in this paper consists of tighter response time bounds for policies that do not make use of explicit job demand information, under more restrictive workload assumptions. The upper bounds in this case hold for a class of policies that we call *processor conserving* policies that do not leave processors idle if jobs can make use of them. Given linear job execution rates (i.e., perfect speedups) and i.i.d. exponential job demands that are independent of all other workload variables, we first derive the optimal and worst case policies. We then show that under the same assumptions the performance of a policy is best when every job can make use of all processors and is worst when all jobs are fully sequential. This leads to computable bounds for the mean response time of any policy in the class that we consider. These results are sensitive to both assumptions of linear execution rates and exponential demands; we give counterexamples to show that the results do not hold for nonexponential demands, thereby implying that *the qualitative behavior of processor allocation policies is sensitive to the assumptions about job demand distribution.*

In addition to the above results, the techniques in this paper, which are based on sample path analysis, may be of interest to the reader. The proof of the main result (lower bound for general workload model)

[1] We define job demand to be the service time of the job on a single processor and available job parallelism to be the maximum number of processors that the job can productively use during its lifetime.

is the simplest. It is based on constructing a uniprocessor policy over every sample path such that every job completes no later in the constructed policy than in the original policy, and then showing that the mean response time of the constructed policy is bounded below by the mean response time of the optimal uniprocessor policy that uses the same information as the original policy. To prove the other results of this paper, suitable coupling between sample paths is required, which leads to more complex proofs than the proof of the main result.

We know of only two previous papers that contain mean response time bounds for parallel processor scheduling policies. First, Leutenegger and Nelson [9] show that PSNTF (Preemptive Smallest Number of Tasks First) is the optimal policy for a set of parallel jobs (no external arrivals) with i.i.d. exponential task service times. The related result in this paper is that PSAPF (Preemptive Smallest Available Parallelism First) is the optimal policy for a system with i.i.d. exponential *job demands* and linear job execution rates up to the job's available parallelism, under a general arrival process. Second, Sevcik [17] shows that for a workload in which each job can make use of *all* processors with perfect speedups (linear execution rates) and job service demands are known to the scheduler, the LWF (Least Work First) policy is optimal when there are no external arrivals, and the LRWF (Least Remaining Work First) is optimal in the case of Poisson arrivals. Also, the LEWF (Least Expected Work First) policy is optimal, either with no arrivals or with Poisson arrivals, if preemption is disallowed and only the expected amount of work is known for each job. As a special case of the main result in Section 3, we show how the counterparts of LRWF and LEWF in the uniprocessor domain (i.e., Shortest Remaining Processing Time and Shortest Expected Processing Time) can be used to obtain lower bounds for two different classes of parallel processor allocation policies.

The remainder of this paper is organized as follows. In Section 2 we present the system model and assumptions. Section 3 derives a lower bound on mean response time under general workload assumptions as well as tighter bounds for specific policies. Section 4 derives optimal and worst case processor allocation policies under i.i.d. exponential job demands that are independent of all other workload variables and under linear job execution rates, given that the policy has no explicit information about job demand. Under the same assumptions computable response time bounds are derived in Section 5 using results for optimal and worst case available parallelism. Sections 3, 4, and 5 each contain experimental data to illustrate the applicability and tightness of the derived bounds under specific demand and parallelism workloads. Finally, Section 6 summarizes the conclusions of this paper.

2 The System Model

Consider an open parallel processor system comprised of P identical processors and a central job queue. A stream of parallel jobs $i = 1, 2, \ldots$ arrive at the system as shown in Figure 1. In this section we define the workload model and the classes of processor allocation policies for the system.

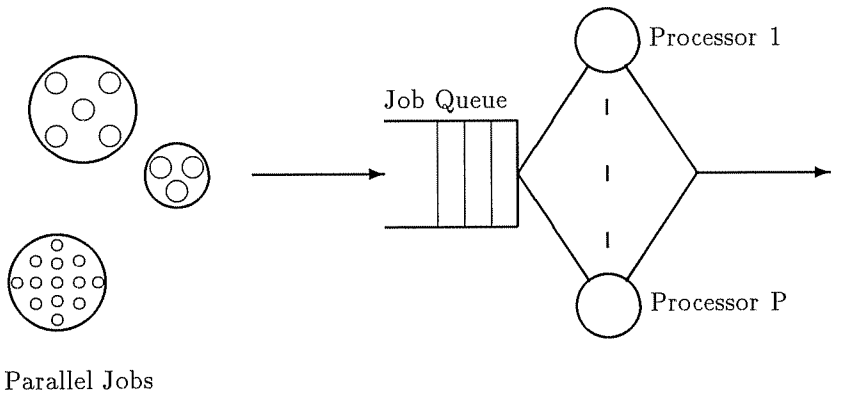


Figure 1: Open System Model

2.1 Workload Model

Each job i has the following five characteristics:

(1) Arrival time A_i,

(2) Total service demand (execution time on one processor) D_i,

(3) Available parallelism $N_i \in \{1, 2, \ldots, P\}$,

(4) Execution rate function $E_i() : [0, P] \rightarrow [0, P]$, $E_i(x) = x$ for $0 \leq x \leq 1$, $E_i(x) \leq x$ and nondecreasing in x for $1 < x \leq N_i$, and $E_i(x) = E_i(N_i)$ for $N_i < x \leq P$.

(5) External class C_i.

The system operates as follows. Upon arrival each job joins the central job queue. At each time $t \geq 0$, the P processors are allocated to jobs present in the queue according to some processor allocation policy

Ψ. If $a_i(t)$ processors are allocated to job i at time t, then its demand is satisfied at rate $E_i(a_i(t))$. Job i leaves the system upon completion of its total demand, D_i. The available parallelism N_i is the maximum number of processors that job i can productively use during its lifetime. That is why $E_i(x) = E_i(N_i)$ for $N_i < x \leq P$.

$\{(A_i, D_i, N_i, E_i(), C_i), i = 1, 2, \ldots\}$ are the primitive workload variables from which other parameters can be computed, e.g., mean demand $\overline{D}$. For the result in Section 3 we allow arbitrary marginal distributions of these primitive variables with arbitrary dependencies between them. That is, we permit any arbitrary joint distribution of these primitive variables. This workload model extends the workload model in [12] by permitting multiple job classes and arbitrary dependencies among the workload variables.

2.2 Processor Allocation Policies

In order to precisely define what we mean by *processor allocation policy* we need the following notation.

$a_i(t)$	=	number of processors or processing power (possibly fractional) allocated to job i at time t.
$\mathcal{A}(t)$	=	set of jobs that have arrived up to time t $(= \{i : A_i \leq t\})$.
X_i	=	departure time of job i $(= \inf\{t \geq 0 : \int_0^t E_i(a_i(s))ds = D_i\})$.
$\mathcal{X}(t)$	=	set of jobs that have departed by time t $(= \{i : X_i \leq t\})$.
$\mathcal{Q}(t)$	=	set of jobs in the system at time t, $\{k_1, k_2, \ldots, k_{\lvert\mathcal{Q}(t)\rvert}\}$ $(= \mathcal{A}(t) - \mathcal{X}(t))$.

Let $I(t)$ be the information available to a processor allocation policy up to time t; that is,

$$I(t) = \left\{|\mathcal{A}(t)|, \{(A_i, D_i, N_i, E_i(), C_i) : i = 1, 2, \ldots, |\mathcal{A}(t)|\}\right\}. \tag{1}$$

We define a processor allocation policy, Ψ, to be a right-continuous process (with left hand limits) that for each time $t \geq 0$ determines how the P processors are allocated amongst the various jobs in the system based on the information available till time t. More precisely, Ψ is a mapping

$$\Psi : (t, I(t)) \rightarrow (a_{k_1}(t), \ldots, a_{k_{|\mathcal{Q}(t)|}}(t)),$$

such that

$$\sum_{i=1}^{|\mathcal{Q}(t)|} a_{k_i}(t) \leq P. \tag{2}$$

Note that processors are only allocated to jobs present in the system, i.e., $a_i(t) = 0$ for $i \notin \mathcal{Q}(t)$. The scheduler may also have external information about jobs in class C_i, e.g., external priority, mean demand, or average parallelism, which is omitted in the above notation.

We define the following classes of processor allocation policies.

- Π is the class of policies as defined above.
- Π_0 is the class of policies where $\{D_i : i \in \mathcal{Q}(t)\}$ are not explicitly known. That is, $I(t)$ is replaced by $I_0(t)$ given by

$$I_0(t) = \left\{|\mathcal{A}(t)|,\ \{(A_i, N_i, E_i(), C_i) : i = 1, 2, \ldots, |\mathcal{A}(t)|\},\ \mathcal{X}(t),\ \{D_i : i \in \mathcal{X}(t)\}\right\}. \tag{3}$$

  Note that $I_0(t)$ is derivable from $I(t)$ (since $\mathcal{X}(t)$ can be obtained from $I(t)$ for all $\Psi \in \Pi$) and hence $\Pi_0 \subset \Pi$. Although $I_0(t)$ contains no explicit information of D_i, $i \in \mathcal{Q}(t)$, it may have partial information about D_i by means of the other primitive variables. For example, C_i may have information about the mean demand, type of demand distribution (e.g., IFR), or distribution of demand (e.g., exponential).

- $\Pi_0^C \subset \Pi_0$ is the class of *processor conserving* policies in Π_0 that have the additional constraint

$$0 \le a_{k_i}(t) \le N_{k_i}, \quad i = 1, 2, \ldots, |\mathcal{Q}(t)|$$

  and also have constraint (2) strengthened to

$$\sum_{i=1}^{|\mathcal{Q}(t)|} a_{k_i}(t) = \min\left(P, \sum_{i=1}^{|\mathcal{Q}(t)|} N_{k_i}\right).$$

  Informally, a processor conserving policy does not allocate more processors to a job than the job can productively make use of, and it does not leave a processor idle if any job can make use of that processor.

In Sections 3 to 6 we will frequently refer to the following policies in Π_0^C:

FCFS: The FCFS policy allocates processors to jobs on a first-come-first-serve basis. Each job is allocated processors as they become available up to a maximum of its available parallelism. Processors released by a departing job are first allocated to the job in service (if any) whose allocation is less than its available parallelism and then to jobs waiting for service.

EQ: The equipartitioning policy allocates an equal fraction of processing power to each job in the system unless a job has smaller available parallelism than the equipartition value, in which case each such job is allocated as many processors as its available parallelism, and the equipartition value is recursively recomputed for the remaining jobs.

PSAPF: Preemptive Smallest Available Parallelism First. The central job queue is a preemptive queue that is ordered in ascending order of available job parallelism. Jobs with the same available parallelism are served in first-come-first-serve order. As in the FCFS policy each job is allocated processors as they become available (or preempted) up to a maximum of its available parallelism, and processors released by a departing job are first allocated to the job in service (if any) whose allocation is less than its available parallelism and then to the jobs waiting for service.

PLAPF: Preemptive Largest Available Parallelism First. Like PSAPF except that the central job queue is ordered in descending order of available job parallelism.

For any given policy Ψ, the mean response time is given by

$$\overline{R}_\Psi := \lim_{n\to\infty} \frac{\sum_{i=1}^{n}(X_i - A_i)}{n}$$

whenever this limit exists. In this paper we are interested in obtaining upper and lower bounds on $\overline{R}_\Psi$ for various classes of policies and in identifying the best and worst policies.

2.3 Additional Notation

For the rest of this paper we use the following notation:

- $\overline{D}$ denotes mean job demand, i.e., $\overline{D} := \lim_{n\to\infty} \frac{\sum_{i=1}^{n} D_i}{n}$ whenever this limit exists.
- $\overline{S}$ denotes mean job service time, i.e., $\overline{S} := \lim_{n\to\infty} \frac{\sum_{i=1}^{n}\left(\frac{D_i}{E_i(N_i)}\right)}{n}$ whenever this limit exists.
- $N \equiv k$ denotes the special case that $N_i = k$, $i = 1, 2, \ldots$, i.e., every job has a constant available parallelism of k, where $1 \le k \le P$.
- $L_{N_i}()$ denotes an execution rate function that is linear up to N_i, i.e., $L_{N_i}(x) = x$ for $0 \le x \le N_i$, and $L_{N_i}(x) = N_i$ for $x < N_i \le P$.
- $L_P()$ denotes a linear execution function up to P, i.e., $L_P(x) = x$, $0 \le x \le P$.
- $G/G/1_P$ denotes a G/G/1 queue with a server of power P.

2.4 Workload Assumptions for Numerical Results

We make the following workload assumptions in the experiments that test the tightness and illustrate the use of the bounds in this paper:

- Jobs arrive according to a Poisson arrival process with rate λ.
- $\{D_i\}_{i=1}^{\infty}$ are i.i.d. and independent of everything else, with mean $\overline{D}$ and coefficient of variation C_D. [2] We use the random variable D to denote total job demand.
- $\{N_i\}_{i=1}^{\infty}$ are i.i.d. and independent of everything else, with mean $\overline{N}$ and coefficient of variation C_N. We use the random variable N to denote available job parallelism.
- Job execution rates are linear up to N, i.e., $E_i() = L_{N_i}()$.

In some cases we consider exponential demands; in other cases we consider hyperexponential demands. In all experiments we set $\overline{D} = P$. (Thus $\rho = \lambda\overline{D}/P = \lambda$.) For most of our experiments we use the following bounded-geometric distribution for available job parallelism (similar to the distribution in [10]), with parameters P_{max} and p

$$N = \begin{cases} P & \text{with probability } P_{max} \\ \min(X, P) & \text{with probability } 1 - P_{max}, \end{cases} \qquad \text{where } X = \text{Geometric}(p).$$

We consider three specific bounded-geometric distributions for N, which are given in Table 1. More details of these workloads are given in [12].

Table 1: Three Bounded-Geometric Distributions for N

Symbol	Parallelism	P_{max}	p	P=20		P=100	
				$\overline{N}$	C_N	$\overline{N}$	C_N
H	High	0.9	1.0	18.10	0.31	90.10	0.33
M	Moderate	0.1	1/(0.4P)	8.70	0.77	43.14	0.80
L	Low	0.1	0.9	3.00	1.89	11.00	2.70

3 Lower Bound for Mean Response Time under General Workload Assumptions

In this section we derive a lower bound on the mean response time per class of parallel processor allocation policies in terms of the mean response time of the optimal uniprocessor policy that uses the same workload information as the class of parallel processor policies and schedules the workload on a uniprocessor of power P (if such an optimal policy exists). The lower bound is attained by first carefully constructing a uniprocessor scheduling policy that allocates at least as much processing power to each unfinished job as the parallel

[2] The coefficient of variation of a random variable is the ratio of the standard deviation to the mean of the random variable.

allocation policy. The mean response time of the constructed policy lower bounds the mean response time of the parallel allocation policy, and is in turn bounded by the optimal uniprocessor policy that uses the same information structure. We present the lower bound result in Section 3.1, and in Section 3.2 we give examples of how to compute the bound for specific cases as well as examples of tighter bounds on a per-policy basis that are suggested by the proof of the lower bound result. We also provide experimental data to illustrate the use of the general lower bound and the tightness of the per-policy bounds.

3.1 The Lower Bound Result

Throughout the remainder of this paper when we speak of a uniprocessor we mean a uniprocessor of power P. That is, the uniprocessor processing power is the same as the total processing power of a parallel system with P identical processors each of unit power. When considering the uniprocessor system, the execution rate for job i when allocated processing power x at any time will be $F_i(x)$, where we assume that $E_i(x) \leq F_i(x) \leq x$, for all $0 \leq x \leq P$. We assume that $F_i()$ is derivable from $E_i()$, N_i, and C_i. Typically, $F_i()$ will be equal to $L_P()$; however, any function that satisfies the given constraints (e.g., $F_i() = E_i()$) is valid. We restrict our attention to the class of nonidling uniprocessor policies Π^1 which allocate the entire processor of power P to the jobs in the system. More precisely,

$$\sum_{i \in \mathcal{Q}^1(t)} a_i^1(t) = P \cdot 1_{(|\mathcal{Q}^1(t)|>0)},$$

where $a_i^1(t)$ and $\mathcal{Q}^1(t)$ for the uniprocessor system have the same interpretation as $a_i(t)$ and $\mathcal{Q}(t)$, respectively, and $1_{(|\mathcal{Q}^1(t)|>0)}$ is the indicator function that there is at least one job in the uniprocessor system at time t. The information structure of Π^1, $I^1(t)$, is the same as $I(t)$ as defined by (1). Note that N_i and $E_i()$ in the information structure of Π^1 no longer have the same interpretation as they did for the parallel processor system. They provide auxiliary information to the uniprocessor scheduler.

We prove the lower bound result by means of Lemma 3.1 and Theorem 3.1 below for the class of policies Π_0, which includes most of the practical parallel processor policies including those that use partial information about service demand characteristics. Lemma 3.1 and Theorem 3.1 also hold for Π and thus the lower bound result is true for policies that have complete knowledge of job demand characteristics (e.g. Preemptive Smallest Cumulative Demand First [11]). In the remark after Theorem 3.1 we comment on the applicability of Lemma 3.1 and Theorem 3.1 for any arbitrary subset $\Pi' \subset \Pi$.

The proof of the lower bound is based on sample path analysis. For a parallel processor policy $\Psi \in \Pi_0$ we construct a uniprocessor policy, $\Psi^1 \in \Pi_0^1 \subset \Pi^1$ such that the response time of each job under Ψ^1 is less than or equal to its response time under Ψ, for every sample path. The information structure of Π_0^1 is given by

$$I_0^1(t) = \big\{|\mathcal{A}(t)|,\ \{(A_i, N_i, E_i(), C_i) : i = 1, 2, \ldots, |\mathcal{A}(t)|\},\ \mathcal{X}^1(t),\ \{D_i : i \in \mathcal{X}^1(t)\}\big\}, \tag{4}$$

$\mathcal{X}^1(t)$ being the set of jobs that have departed the uniprocessor system by time t. The key feature of Ψ^1 is that it allocates at least as much processing power to each unfinished job as Ψ does. This construction is given in Lemma 3.1 below. The mean response time of the constructed uniprocessor policy is in turn bounded below by the mean response time of the optimal policy in Π_0^1 which leads to the lower bound in Theorem 3.1.

Lemma 3.1 *Given any parallel processor policy $\Psi \in \Pi_0$, there exists a uniprocessor policy $\Psi^1 \in \Pi_0^1$ such that*

$$a_i^1(t) \geq a_i(t), \quad i \in \mathcal{Q}^1(t), \quad \forall t \geq 0, \tag{5}$$

where $a_i^1(t)$ and $a_i(t)$ are the processor allocations to job i at time t under policies Ψ^1 and Ψ, respectively. Moreover, for any such policy Ψ^1,

$$\overline{R}_{\Psi} \geq \max(\overline{S},\ \overline{R}_{\Psi^1}).$$

Proof. The proof of this lemma is established pathwise by letting the uniprocessor and parallel processor systems have the same primitive workload variables $\{(A_i, D_i, N_i, E_i(), C_i), i = 1, 2, \ldots\}$. Consider the uniprocessor policy Ψ^1 obtained by

$$a_i^1(t) = a_i(t) + \frac{P - \sum_{i \in \mathcal{Q}^1(t)} a_i(t)}{|\mathcal{Q}^1(t)|} \geq a_i(t) \quad \text{for } i \in \mathcal{Q}^1(t), \quad |\mathcal{Q}^1(t)| > 0. \tag{6}$$

Clearly Ψ^1 satisfies (5) and $\sum_{i \in \mathcal{Q}^1(t)} a_i^1(t) = P \cdot 1_{(|\mathcal{Q}^1(t)| > 0)}$. In order to establish that $\Psi^1 \in \Pi_0^1$ it remains to verify that $\Psi^1(t) = (a_i^1(t) : i \in \mathcal{Q}^1(t))$ defined as above is a valid policy, i.e., it is a function of $I_0^1(t)$ given by (4). The difficulty is that $a_i^1(t)$ is defined in terms of $a_i(t)$ which in turn is a function of $I_0(t)$ given by (3). If we can establish that $\mathcal{X}^1(t) \supseteq \mathcal{X}(t)$ then we can show that $I_0(t)$ is derivable from $I_0^1(t)$, and consequently $\Psi^1(t)$ will be a function of $I_0^1(t)$ as required[3].

[3] To show that $I_0(t)$ is derivable from $I_0^1(t)$ if $\mathcal{X}^1(t) \supseteq \mathcal{X}(t)$ we need only show that $\mathcal{X}(t)$ is derivable from $I_0^1(t)$. This can be done by simulating Ψ from $t = 0$ onwards (note that $\{D_i : i \in \mathcal{X}^1(t)\}$ is known to the simulator) and progressively constructing $\mathcal{X}(s)$, $0 \leq s \leq t$, in the simulation, by checking if $\int_0^s E_i(a_i(u))du = D_i$, $i \in \mathcal{X}^1(t)$.

We now show that any uniprocessor policy that satisfies (5) also satisfies $\mathcal{X}^1(t) \supseteq \mathcal{X}(t)$ or equivalently $\mathcal{Q}^1(t) \subseteq \mathcal{Q}(t)$ (since $\mathcal{A}^1(t) = \mathcal{A}(t)$). Consider any $i \in \mathcal{Q}^1(t)$. By (5) we have that $a_i^1(s) \geq a_i(s)$ for $s \in [A_i, t]$. Hence,

$$D_i > \int_0^t F_i(a_i^1(s))ds = \int_{A_i}^t F_i(a_i^1(s))ds \geq \int_{A_i}^t E_i(a_i^1(s))ds \geq \int_{A_i}^t E_i(a_i(s))ds = \int_0^t E_i(a_i(s))ds, \quad i \in \mathcal{Q}^1(t), \tag{7}$$

where we used the fact that $F_i() \geq E_i()$ and $E_i()$ is nondecreasing. Since $i \in \mathcal{Q}^1(t)$ has not departed by time t under Ψ^1 it follows from (7) that it has not departed under Ψ either. Hence $i \in \mathcal{Q}(t)$, and therefore $\mathcal{Q}^1(t) \subseteq \mathcal{Q}(t)$, as required.

Finally, note that $\mathcal{X}^1(t) \supseteq \mathcal{X}(t)$, $t \geq 0$, implies that $X_i^1 \leq X_i$, $i = 1, 2, \ldots$. Therefore, $R_i^1 = X_i^1 - A_i \leq X_i - A_i = R_i$, $i = 1, 2, \ldots$, which implies that $\overline{R}_{\Psi^1} \leq \overline{R}_\Psi$. The lower bound of the lemma follows since $\overline{R}_\Psi \geq \overline{S}$. ∎

Lemma 3.1 leads to the following lower bound:

Theorem 3.1 *For any parallel processor policy $\Psi \in \Pi_0$,*

$$\overline{R}_\Psi \geq \max\left\{\overline{S}, \inf_{\Psi^1 \in \Pi_0^1} \overline{R}_{\Psi^1}\right\} \geq \max\left\{\overline{S}, \inf_{\Psi^1 \in \Pi_0^1} \overline{R}_{\Psi^1}[F_i() = L_P()]\right\},$$

where $L_P(x) = x$, $0 \leq x \leq P$, denotes the linear execution rate function up to P.

Proof. The first lower bound follows because $\inf_{\Psi^1 \in \Pi_0^1} \overline{R}_{\Psi^1}$ is less than or equal to the bound in Lemma 3.1. The second lower bound follows because $F_i(x) \leq L_P(x)$, $0 \leq x \leq P$. ∎

Remark: While Lemma 3.1 and Theorem 3.1 have been established for parallel processor policies $\Psi \in \Pi_0$, these results can be extended to classes $\Pi' \subseteq \Pi$ that have other "partial" information structures (i.e., subsets of $I(t)$). The key step is to show that there exists a uniprocessor policy $\Psi^1 \in \Pi^{1\prime} = \Pi^1 \cap \Pi'$ satisfying (5). This can be done by appropriately modifying Ψ as done in Lemma 3.1. However, the main hurdle in showing that $\Psi^1 \in \Pi^{1\prime}$ is to show that $I'(t)$ is obtainable from $I^{1\prime}(t)$, i.e., at any given time Ψ^1 has at least as much information as Ψ does[4]. This verification is quite straightforward for several different partial information

[4]For example, only partial information about $E_i()$ might be available in $I'(t)$, say only $E_i(1) = 1$. In this case a parallel

structures. In particular for the case of complete information

$$I^1(t) = I(t) = \{|\mathcal{A}(t)|, \{(A_i, D_i, N_i, E_i(), C_i) : i = 1, 2, \ldots, |\mathcal{A}(t)|\}\},$$

and Lemma 3.1 and Theorem 3.1 continue to hold.

3.2 Applications of the Lower Bound Result

In this section we give applications of Theorem 3.1 by first examining how known optimality results for uniprocessor policies can be used to bound the mean response time of particular classes of parallel processor policies. We then give experimental results to illustrate how the general lower bound can be used as a reference point for comparing the performance of parallel processor policies. In Section 3.2.3 we then examine cases where tighter bounds can be obtained on a per-policy basis directly from Lemma 3.1 and in Section 3.2.4 we present experimental data to illustrate their tightness. Since uniprocessor results have been obtained for linear job execution rates, the applications in this section assume that $F_i() = L_P()$, $i = 1, 2, \ldots$.

3.2.1 Optimal Policy Bounds

From Theorem 3.1 we observe that the same lower bound holds for all parallel processor policies that share the same information structure (e.g., Π, Π_0). Consider, any $\Psi \in \Pi$. It is well known that the Shortest Remaining Processing Time (SRPT) policy is optimal over all uniprocessor policies [2,14]. As a result,

$$\overline{R}_\Psi \geq \max(\overline{S}, \overline{R}^1_{SRPT}), \quad \forall \Psi \in \Pi.$$

We use the superscript 1 to denote that the policy is a uniprocessor policy. For Poisson job arrivals we can compute $\overline{R}^1_{SRPT}$ from the analysis in [13].

The SRPT policy uses complete knowledge of job demands. We can obtain tighter bounds if only restricted job demand information is available to the scheduler, e.g., policies in Π_0. First consider cases where $\{D_i\}_{i=1}^{\infty}$ are i.i.d. and independent of everything else. If D_i has an increasing failure rate[5] (IFR) distribution then the FCFS policy is optimal in Π^1_0 [5], and if D_i has a decreasing failure rate (DFR) distribution then the Foreground-Background (FB) policy is optimal in Π^1_0 [8,15]. (The FB policy gives

processor policy can allocate one processor continuously to job i and thus obtain D_i when job i exits. However, since all uniprocessor policies being considered are nonidling, it may not be possible to ensure that $a^1_i(t) = 1$ for all time until job i exits in which case D_i will not be known to the uniprocessor scheduler even after job i departs.

[5] A nondiscrete distribution F has increasing (decreasing) failure rate iff for any $\epsilon > 0$ we have that $(F(t+\epsilon) - F(t))/(1 - F(t))$ increases (decreases) with t for all $t > 0$ and for $1 - F(t) > 0$ [8].

highest priority to the job that has attained least amount of service and processor shares only jobs of the highest priority.) Thus we have,

$$\overline{R}_\Psi \geq \max(\overline{S}, \overline{R}^1_{FCFS}), \quad \forall \Psi \in \Pi_0, \quad D_i \sim IFR,$$
$$\overline{R}_\Psi \geq \max(\overline{S}, \overline{R}^1_{FB}), \quad \forall \Psi \in \Pi_0, \quad D_i \sim DFR.$$

Next, consider cases where D_i is exponentially distributed with rate μ_i, where μ_i is known to the scheduler, $i = 1, 2, \ldots$. The rate μ_i is known either through external information about C_i, or because it is an explicit function of N_i (e.g., mean demand is positively correlated with available parallelism through some known function). We assume that the scheduler has no other information about D_i upon arrival of job i. For such a workload the preemptive Shortest Expected Processing Time first (SEPT) policy is optimal in Π^1_0 [8,15], and therefore

$$\overline{R}_\Psi \geq \max(\overline{S}, \overline{R}^1_{SEPT}), \quad \forall \Psi \in \Pi_0, \quad D_i \sim \exp(\mu_i).$$

3.2.2 Application of the Optimal Policy Bounds

The general lower bound provides an upper bound on the ratio of the mean response time for a given policy and the best achievable performance. To illustrate how the general lower bound is useful as a reference point for policy comparison, we assume that $\{D_i\}_{i=1}^{\infty}$ are i.i.d. and independent of everything else and let D denote the random variable for job demand with mean $\overline{D} = P$ and coefficient of variation C_D. We compare the PSAPF, FCFS, and EQ policies in Π_0 assuming a Poisson job arrival process with rate λ, a two-stage hyperexponential (H_2) distribution for D, and linear job execution rates ($E_i() = L_{N_i}()$ and $F_i() = L_P()$). Since the H_2 distribution is DFR the general lower bound is given by $\overline{R}^*_1 \equiv \max(\overline{S}, \overline{R}^1_{FB})$. $\overline{R}_{PSAPF}$, $\overline{R}_{FCFS}$, and $\overline{R}_{EQ}$ were estimated by discrete event simulation[6], and $\overline{R}^1_{FB}$ was obtained from the analysis in [8].

Figure 2 plots the ratio $\overline{R}_\Psi / \overline{R}^*_1$ versus $\rho = \lambda \overline{D}/P$ for the H, M, and L parallelism distributions given in Table 1 and P=20. Note that the Y-axis of the figure has a log scale. We observe the following:

- EQ has a significantly lower response time ratio than FCFS and PSAPF for each workload.
- PSAPF is not the optimal policy for this job demand distribution (it is optimal for exponential demands as will be shown in Section 4).

[6] All simulation experiments in this paper have 95% confidence intervals with less than 5% half-widths. We used either the regenerative method of simulation or the method of batch means, depending on how time-consuming it was to obtain regenerative cycles.

- $\overline{R}_{EQ}$ is less than twice the lower bound for all three parallelism workloads, whereas the ratios for FCFS and PSAPF are as high as 16. For the given workloads, the distance between the EQ policy and the best achievable performance in Π_0 is guaranteed to be less than or equal to the observed ratios for $\overline{R}_{EQ}/\overline{R}_1^*$.

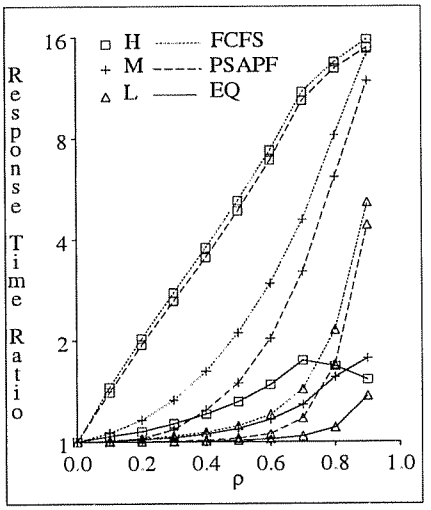


Figure 2: $\overline{R}_\Psi/\overline{R}_1^*$, $\Psi \in \{PSAPF, FCFS, EQ\}$
$P = 20$, $D \sim H_2$, $C_D = 5$

3.2.3 Tighter Per-Policy Bounds

In the general case, a policy Ψ^1 that satisfies the constraints of Lemma 3.1 with respect to a parallel processor policy Ψ may be (nearly) as complex to analyze as Ψ. However, for certain parallel processor policies or under certain workloads we can obtain a Ψ^1 that is either easy to analyze or the analysis is already known, and thus it is possible to compute the mean response time bound of Lemma 3.1. We give two examples to show how known uniprocessor policies can be used directly in Lemma 3.1 to provide a tighter bound than Theorem 3.1 for specific parallel processor policies.

Consider $\Psi = RRP \in \Pi_0^C$, where RRP stands for the Round-Robin-Process policy [11]. Under RRP there is a global queue of processes and all processes are served in round-robin order. This is like the process scheduling mechanism in the Sequent multiprocessor if we assume that all processes have the same priority [19]. When the quantum size goes to zero the processor allocation to each job is directly proportional

to its available parallelism and is given by (see [9]),

$$a_i(t) = \min\left(N_i, \frac{N_i}{\sum_{j\in\mathcal{Q}(t)} N_j} P\right), \quad i \in \mathcal{Q}(t).$$

It is easy to verify that $RRP \in \Pi_0^C$.[7]Now consider $\Psi^1 = DPS \in \Pi_0^1$, where DPS stands for the Discriminatory Processor Sharing policy [6]. The allocation of processing power under DPS is given by

$$a_i^1(t) = \frac{g_i}{\sum_{j\in\mathcal{Q}^1(t)} g_j} P, \quad i \in \mathcal{Q}^1(t),$$

where $0 < g_i < \infty$ is the discrimination weight of job i. If we set the discrimination weight $g_i = N_i$, we obtain

$$\begin{aligned} a_i^1(t) &= \frac{N_i}{\sum_{j\in\mathcal{Q}^1(t)} N_j} P, \quad i \in \mathcal{Q}^1(t) \\ &\geq \min\left(N_i, \frac{N_i}{\sum_{j\in\mathcal{Q}(t)} N_j} P\right) \quad \text{if } \mathcal{Q}^1(t) \subseteq \mathcal{Q}(t) \\ &= a_i(t). \end{aligned}$$

By induction over time it is straightforward to obtain that $\mathcal{Q}^1(t) \subseteq \mathcal{Q}(t)$ and thus $a_i^1(t) \geq a_i(t)$, $i \in \mathcal{Q}^1(t)$. Using Lemma 3.1 we obtain

$$\overline{R}_{RRP} \geq \max(\overline{S}, \overline{R}^1_{DPS}(g_i = N_i)).$$

$\overline{R}^1_{DPS}$ for arbitrary g_i's can be obtained assuming a Poisson arrival process from the analysis in [3].

As the second example, consider $\Psi = EQ \in \Pi_0^C$ and a workload with $N_i \geq P/2$, $i = 1, 2, \ldots$. For this workload the processor allocation under EQ is given by

$$a_i(t) = \min\left(N_i, \frac{P}{|\mathcal{Q}(t)|}\right), \quad i \in \mathcal{Q}(t).$$

[7]To verify that $RRP \in \Pi_0^C$ we observe that if $\sum_{j\in\mathcal{Q}(t)} N_j \leq P$ then $a_i(t) = N_i$, for all $i \in \mathcal{Q}(t)$, and thus $\sum_{i\in\mathcal{Q}(t)} a_i(t) = \sum_{i\in\mathcal{Q}(t)} N_i$; whereas if $\sum_{j\in\mathcal{Q}(t)} N_j > P$ then $a_i(t) = N_i P / \sum_{j\in\mathcal{Q}(t)} N_j$, for all $i \in \mathcal{Q}(t)$, and thus $\sum_{i\in\mathcal{Q}(t)} a_i(t) = P$.

It is easy to verify that this allocation is processor conserving. Now consider the Processor Sharing (PS) policy in Π_0^1, in which the allocation of processing power is given by

$$a_i^1(t) = \frac{P}{|\mathcal{Q}^1(t)|}, \quad i \in \mathcal{Q}^1(t).$$

Clearly $a_i^1(t) \geq a_i(t)$ if $|\mathcal{Q}^1(t)| \leq |\mathcal{Q}(t)|$ which can easily be shown by induction over time. Thus Lemma 3.1 yields

$$\overline{R}_{EQ}(N \geq P/2) \geq \max(\overline{S}, \overline{R}_{PS}^1),$$

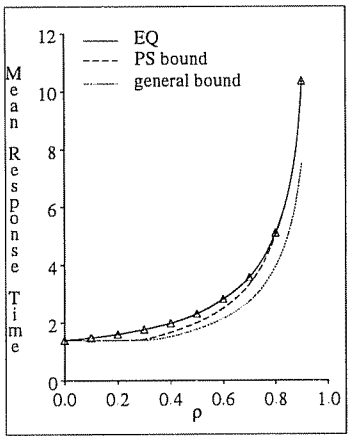
where $N \geq P/2$ denotes $N_i \geq P/2$, $i = 1, 2, \ldots$.

3.2.4 Tightness of the Per-Policy Bounds

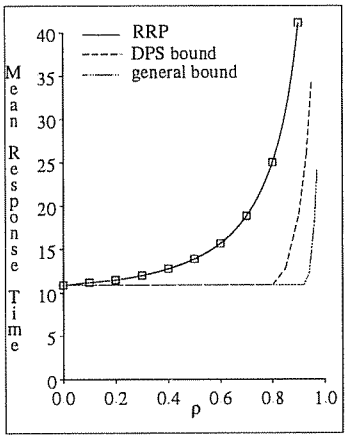
To test whether the per-policy bounds for the EQ($N \geq P/2$) and RRP policies are significantly tighter than the general lower bound, we make the same assumptions as in the experiment of Figure 2, except that we set P=100 and for the EQ policy we let N =Uniform[50,100] so that $N \geq P/2$. $\overline{R}_{EQ}$ was estimated using simulation and $\overline{R}_{PS}^1$ is the mean response time of an $M/G/1_P$ PS queue, which equals $(\overline{D}/P)/(1-\rho)$ (see [8]). $\overline{R}_{RRP}$ was estimated using discrete event simulation and $\overline{R}_{DPS}^1$ was computed using the analysis in [4] (which shows how the analysis of [3] can be simplified for hyperexponential demand distributions).

First consider the EQ policy, where $\overline{R}_{EQ}(N \geq P/2) \geq \max(\overline{S}, \overline{R}_{PS}^1)$. Figure 3a plots $\overline{R}_{EQ}$, the PS bound and the general lower bound for the stated workload. From Figure 3a we observe that the PS bound is appreciably better than the general bound at utilizations above 0.5. At very low utilizations the PS bound is simply $\overline{S}$ and is thus equal to the general bound. However, at high utilizations $\overline{R}_{PS}^1$ is very close to $\overline{R}_{EQ}$ because as shown above the EQ($N \geq P/2$) policy reduces to the PS policy when there are two or more jobs in the system (which is very likely at high utilizations). Thus, for this workload the PS bound serves as an accurate estimate for $\overline{R}_{EQ}$ at very high utilizations. (We have observed similar results for other workloads with $N \geq P/2$.)

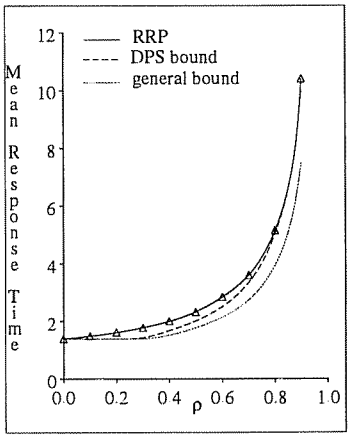
Now consider the RRP policy, where $\overline{R}_{RRP} \geq \max(\overline{S}, \overline{R}_{DPS}^1)$, the job discrimination weight in DPS being equal to its available parallelism. In addition to the Uniform[50,100] parallelism workload we also consider the H and M workloads for N. (Recall that the DPS bound for RRP holds for all distributions of N, not just $N \geq P/2$.) Figure 3b displays $\overline{R}_{RRP}$, the DPS bound and the general lower bound for the H workload,



(a) EQ: N=Uniform[50,100]



(b) RRP: H workload



(c) RRP: N=Uniform[50,100]

Figure 3: Per-Policy Bounds

$P = 100$, $D \sim H_2$, $C_D = 5$

and Figure 3c presents the same results for the Uniform[50,100] workload. Comparing Figures 3b and 3c we observe that the DPS bound is considerably tighter for the Uniform[50,100] workload than for the H workload. The DPS bound is looser for the M workload (not shown) than for the H workload. The reason for the looseness of the bound under the H and M workloads is that processors are idle under RRP if there are only a few jobs in the system, each with low parallelism. As shown in Table 1, 10% of the H workload consists of sequential jobs (N=1). For the M workload it can be shown that slightly more than 10% of the jobs have an available parallelism of 5 or less. Low parallelism does not affect the processor utilization under DPS because whenever there is one or more job in the system DPS achieves 100% processor utilization. This causes $\overline{R}_{DPS}$ to be significantly lower than $\overline{R}_{RRP}$ for the H and M workloads. On the other hand $\overline{R}_{DPS}$ is much tighter for the N =Uniform[50,100] workload because all processors are fully utilized under RRP for this workload if there are two or more jobs in the system (every job has an available parallelism of at least P/2). The above suggests that uniprocessor bounds may not be tight for workloads that have a substantial fraction of jobs with low parallelism. We note, however, that these experiments have assumed linear execution rates. For sublinear execution rates the bounding uniprocessor policy will not efficiently utilize the entire processing power, particularly when there are very few jobs in the system. This could improve the tightness of the uniprocessor bounds.

4 Response Time Bounds for Exponential Demands: Optimal and Worst Case Policies

In Section 3 we derived a general lower bound for parallel processor policies. We saw that this bound is quite loose for certain policies under specific workloads. This motivates the search for tighter bounds under specific workloads. In this section and in section 5 we consider policies in Π_0 (policies that do not have explicit knowledge of job demands) and we derive tighter bounds under the following set of workload assumptions:

A1. The total service demands $\{D_i\}_{i=1}^{\infty}$ are i.i.d. exponential with parameter $\mu = 1/\overline{D}$, and are independent of everything else.

A2. The execution rate function of job i is linear up to N_i, i.e., $E_i(x) = L_{N_i}(x)$, $0 \leq x \leq P$, $i = 1, 2, \ldots$.

In Section 4.1 we determine the processor allocation policy in Π_0 that has the smallest mean response time under assumptions A1 and A2, and also determine the processor allocation policy in Π_0^C that has the highest mean response time under these assumptions. We thus obtain achievable lower and upper bounds on mean response time for all policies in Π_0 and Π_0^C, respectively. Section 4.2 gives applications of the PSAPF and PLAPF bounds under assumptions A1 and A2. In Section 4.3 we provide counterexamples to show that these bounds do not hold when assumptions A1 and A2 are violated.

4.1 Optimal and Worst Case Policies Under A1 and A2

To motivate the optimal and worst case policies under assumptions A1 and A2, consider the behavior of a system Γ_Ψ with processor allocation policy $\Psi \in \Pi_0^C$. If a job in Γ_Ψ is allocated processing power 'a' then its residual lifetime is exponentially distributed with rate $a\mu$. If there are $k > 0$ jobs in Γ_Ψ with the j^{th} job having a processor allocation of a_j then the time to the next departure from Γ_Ψ is exponentially distributed with rate $\sum_{j=1}^{k} a_j \mu$. Therefore, as long as all processors are busy in Γ_Ψ, departures occur with rate $P\mu$. At other times the departure rate of jobs is less than $P\mu$.

Let us compare two processor allocation policies Ψ_1 and Ψ_2, $\Psi_1, \Psi_2 \in \Pi_0^C$, under assumptions A1 and A2. If we start out with zero jobs in the system, Ψ_1 and Ψ_2 have the same behavior until all processors are occupied. Once all processors are occupied job departures occur at the same rate of $P\mu$ under each of Ψ_1 and Ψ_2. Departures continue to occur at rate $P\mu$ in both systems until the total processor utilization drops

to less than P under one of Ψ_1 or Ψ_2. From this point onwards the departure rate of jobs need not be the same under Ψ_1 and Ψ_2 (unless job arrivals cause all processors to be busy once again). In fact, the departure rate of jobs is higher for the system that has a higher number of processors busy. Thus whenever a queue builds up it is better for overall policy performance to serve the jobs with high amounts of parallelism last, to keep as many processors busy as possible.

Given this background we now show that under assumptions A1 and A2 the PSAPF policy defined in Section 2.2 performs better than any other policy $\Psi \in \Pi_0$, and likewise the PLAPF policy performs worse than any other policy $\Psi \in \Pi_0^C$. PSAPF assigns low priority to jobs with high parallelism. Thus whenever there are many jobs in the system (such that all processors are busy) jobs that have low available parallelism are likely to depart under PSAPF whereas jobs with high available parallelism are held for later. Once the number of jobs in the system falls down to a small value, the jobs with high available parallelism begin execution and thereby keep more processors busy as compared to any other policy in Π_0. We prove the optimality of PSAPF by means of the Theorem 4.1 below. The proof of this theorem is not as simple as the proof of the general lower bound. Readers who wish to skip this proof on first reading can move on to Theorem 4.2 (which shows that PLAPF is pessimal in Π_0^C) without loss of continuity.

Theorem 4.1 *Under assumptions A1 and A2,*

$$\overline{R}_{\Psi} \geq \overline{R}_{PSAPF}, \quad \forall \Psi \in \Pi_0.$$

Proof. Let Γ_{PSAPF} be a system with the PSAPF policy and Γ_{Ψ} be a system with any other policy $\Psi \in \Pi_0$. Let $Q_{\Theta}(t)$ equal the number of jobs in Γ_{Θ} at time t, $\Theta \in \{PSAPF, \Psi\}$. We prove this theorem by suitably coupling sample paths for Γ_{PSAPF} and Γ_{Ψ}, and showing that for every sample path $Q_{PSAPF}(t) \leq Q_{\Psi}(t)$, $\forall\, t$, from which it will follow that $\overline{R}_{PSAPF} \leq \overline{R}_{\Psi}$.

For the purposes of the proof we assume that there is a list of jobs in Γ_{Θ} at time t, distinct from the job queue, sorted in increasing order of available parallelism[8], with jobs of the same available parallelism in FCFS order, $\Theta \in \{PSAPF, \Psi\}$. Let $\sigma_i^{\Theta}(t)$ denote the job in position i of Γ_{Θ}'s list at time t, $i = 1, 2, \ldots, Q_{\Theta}(t)$, $\Theta \in \{PSAPF, \Psi\}$. To provide less cumbersome notation let $M_i^{\Theta}(t) \equiv N_{\sigma_i^{\Theta}(t)}$ denote the available parallelism and $b_i^{\Theta}(t) \equiv a^{\Theta}_{\sigma_i^{\Theta}(t)}(t)$ the processor allocation at time t of job $\sigma_i^{\Theta}(t)$. Define $M_0^{\Theta}(t) := 0$ and $M_i^{\Theta}(t) := \infty$

[8] Although this corresponds to queue ordering under PSAPF, it need not correspond to queue ordering under Ψ.

for $i > Q_\Theta(t)$. These boundary values will be useful during list insertions. Let

$$\alpha_i^\Theta(t) := \frac{\min\left(b_i^\Theta(t), M_i^\Theta(t)\right)}{P}, \quad i = 1, \ldots, Q_\Theta(t), \quad \Theta \in \{PSAPF, \Psi\}.$$

Thus, job $\sigma_i^\Theta(t)$ completes at rate $\alpha_i^\Theta(t)\, P\mu$ from Γ_Θ. Note that $b_i^\Theta(t) \leq M_i^\Theta(t)$ for $\Theta \in \Pi_0^C$, but this is not the case for every policy in Π_0 (e.g., a fixed partitioning policy that allocates more than N_j processors to job j).

Coupling of Sample Paths in Γ_{PSAPF} and Γ_Ψ

Fix $\{A_j, N_j, C_j\}_{j=1}^\infty$ as the same for both Γ_{PSAPF} and Γ_Ψ. Fixing N_j automatically determines $E_j()$, $j = 1, 2, \ldots$ since execution rates are linear up to N_j. Consider that *potential job completions* [20] occur at jumps of a Poisson process with rate $P\mu$. Fix the same potential completion instants $\{T_j\}_{j=1}^\infty$ in both Γ_{PSAPF} and Γ_Ψ. To generate *actual* job completion times in Γ_{PSAPF} and Γ_Ψ let $\{U_j\}_{j=1}^\infty$ be i.i.d. Uniform[0,1) random variables. At the j^{th} potential completion instant T_j, the job in position k of Γ_Θ's list (i.e., job $\sigma_k^\Theta(T_j)$) departs from Γ_Θ if

$$U_j \in \left[\sum_{\ell=1}^{k-1} \alpha_\ell^\Theta(T_j^-),\ \sum_{\ell=1}^{k} \alpha_\ell^\Theta(T_j^-)\right), \quad \Theta \in \{PSAPF, \Psi\}. \tag{8}$$

This ensures that the probability that $\sigma_k^\Theta(T_j)$ departs from Γ_Θ is $\alpha_k^\Theta(T_j^-)$, $\Theta \in \{PSAPF, \Psi\}$.

Sample Path Analysis

Using the above coupling of sample paths we show by induction over time that for every pair of coupled sample paths, for all $t \geq 0$

$$Q_{PSAPF}(t) \leq Q_\Psi(t), \quad \text{and} \quad M_i^{PSAPF}(t) \geq M_i^\Psi(t), \quad i = 1, \ldots, Q_{PSAPF}(t). \tag{9}$$

The second inequality implies that

$$\alpha_i^{PSAPF}(t) = \frac{M_i^{PSAPF}(t)}{P} \geq \frac{\min\left(b_i^\Psi(t),\, M_i^\Psi(t)\right)}{P} = \alpha_i^\Psi(t), \quad i = 1, 2, \ldots, n(t), \tag{10}$$

where $n(t) = \max\{j = 1, \ldots, Q_{PSAPF}(t) \ :\ b_j^{PSAPF}(t) = M_j^{PSAPF}(t)\} \geq 1$ when $Q_{PSAPF}(t) > 0$. Note that $\sigma_{n(t)}^{PSAPF}(t)$ is the job with the maximum index in Γ_{PSAPF}'s list at time t to receive as many processors as its available parallelism.

We carry out the induction over arrival instants and potential completion instants since no jobs depart in between these event times. Let $\{t_i\}_{i=0}^{\infty}$ be the sequence of arrival and potential completion times arranged in increasing order. Let both Γ_{PSAPF} and Γ_{Ψ} start out with zero jobs each. Then clearly (9) is satisfied at $t = t_0$. Assume that (9) is true for all $t \leq t_j$. Since no jobs arrive or depart in (t_j, t_{j+1}) (9) is satisfied for all $t < t_{j+1}$. We now prove that (9) is true at $t = t_{j+1}$. Consider all possible events at time t_{j+1}.

1. Arrival of Job m:

   By the induction hypothesis it follows that

   $$Q_{PSAPF}(t_{j+1}) = Q_{PSAPF}(t_j) + 1 \leq Q_{\Psi}(t_j) + 1 = Q_{\Psi}(t_{j+1}).$$

   Let k be the position in Γ_{PSAPF}'s list where job m is inserted at t_{j+1} (i.e., $m = \sigma_k^{PSAPF}(t_{j+1})$). Then $M_{k-1}^{PSAPF}(t_j) \leq N_m < M_k^{PSAPF}(t_j)$. (Recall that in the sorted list jobs with the same available parallelism are maintained in FCFS order.) Since $M_{k-1}^{\Psi}(t_j) \leq M_{k-1}^{PSAPF}(t_j) \leq N_m$ it follows that $m = \sigma_{\ell}^{\Psi}(t_{j+1})$ for some $\ell \geq k$. Thus,

   $$\begin{aligned}
   M_i^{PSAPF}(t_{j+1}) = M_i^{PSAPF}(t_j) \geq M_i^{\Psi}(t_j) = M_i^{\Psi}(t_{j+1}), &\qquad i = 1, \ldots, k-1, \\
   M_i^{PSAPF}(t_{j+1}) \geq N_m \geq M_i^{\Psi}(t_{j+1}), &\qquad i = k, \ldots, \ell, \\
   M_i^{PSAPF}(t_{j+1}) = M_{i-1}^{PSAPF}(t_j) \geq M_{i-1}^{\Psi}(t_j) = M_i^{\Psi}(t_{j+1}), &\qquad i = \ell+1, \ldots, Q_{PSAPF}(t_{j+1}).
   \end{aligned}$$

2. Potential Completion:

   (a) No departure from each of Γ_{PSAPF} and Γ_{Ψ}: The induction hypothesis continues to be true at $t = t_{j+1}$.

   (b) Departure from Γ_{PSAPF} but not from Γ_{Ψ}:

   $$Q_{PSAPF}(t_{j+1}) = Q_{PSAPF}(t_j) - 1 \leq Q_{\Psi}(t_j) - 1 = Q_{\Psi}(t_{j+1}) - 1 < Q_{\Psi}(t_{j+1}).$$

   Suppose $\sigma_k^{PSAPF}(t_j)$ departs from Γ_{PSAPF}. Then

   $$\begin{aligned}
   M_i^{PSAPF}(t_{j+1}) = M_i^{PSAPF}(t_j) \geq M_i^{\Psi}(t_j) = M_i^{\Psi}(t_{j+1}), &\qquad i = 1, \ldots, k-1, \\
   M_i^{PSAPF}(t_{j+1}) = M_{i+1}^{PSAPF}(t_j) \geq M_i^{PSAPF}(t_j) \geq M_i^{\Psi}(t_j) = M_i^{\Psi}(t_{j+1}), &\qquad i = k, \ldots, Q_{PSAPF}(t_{j+1}).
   \end{aligned}$$

   (c) Departure from each of Γ_{PSAPF} and Γ_{Ψ}:

   $$Q_{PSAPF}(t_{j+1}) = Q_{PSAPF}(t_j) - 1 \leq Q_{\Psi}(t_j) - 1 = Q_{\Psi}(t_{j+1}).$$

Let $\sigma_k^{PSAPF}(t_{j+1})$ and $\sigma_m^{\Psi}(t_{j+1})$ depart from Γ_{PSAPF} and Γ_{Ψ}, respectively. Consider the following two cases:

(i) $k \leq m$:

$$M_i^{PSAPF}(t_{j+1}) = M_i^{PSAPF}(t_j) \geq M_i^{\Psi}(t_j) = M_i^{\Psi}(t_{j+1}), \quad i = 1, \ldots, k-1$$

$$M_i^{PSAPF}(t_{j+1}) = M_{i+1}^{PSAPF}(t_j) \geq M_i^{PSAPF}(t_j) \geq M_i^{\Psi}(t_j) = M_i^{\Psi}(t_{j+1}), \quad i = k, \ldots, m-1$$

$$M_i^{PSAPF}(t_{j+1}) = M_{i+1}^{PSAPF}(t_j) \geq M_{i+1}^{\Psi}(t_j) = M_i^{\Psi}(t_{j+1}), \quad i = m, \ldots, Q_{PSAPF}(t_{j+1})$$

(ii) $k > m$:

This case is infeasible. To see this assume that $k > m$. Let $t_{j+1} = T_r$, the r^{th} potential completion instant, $1 \leq r < j+1$. From (8) it follows that

$$U_r \in I_1 \quad \text{and} \quad U_r \in I_2, \tag{11}$$

where

$$I_1 = \left[\sum_{\ell=1}^{m-1} \alpha_\ell^{\Psi}(t_{j+1}^-),\ \sum_{\ell=1}^{m} \alpha_\ell^{\Psi}(t_{j+1}^-)\right) \quad \text{and} \quad I_2 = \left[\sum_{\ell=1}^{k-1} \alpha_\ell^{PSAPF}(t_{j+1}^-),\ \sum_{\ell=1}^{k} \alpha_\ell^{PSAPF}(t_{j+1}^-)\right).$$

Since $\sigma_k^{PSAPF}(t_{j+1}^-)$ departs from Γ_{PSAPF}, $k-1 \leq n(t_{j+1}^-) = \max\{j : b_j^{PSAPF}(t_{j+1}^-) = M_j^{PSAPF}(t_{j+1}^-)\}$. Applying (10) we have,

$$\sum_{\ell=1}^{m} \alpha_\ell^{\Psi}(t_{j+1}^-) \leq \sum_{\ell=1}^{m} \alpha_\ell^{PSAPF}(t_{j+1}^-) \leq \sum_{\ell=1}^{k-1} \alpha_\ell^{PSAPF}(t_{j+1}^-),$$

which means that intervals I_1 and I_2 are disjoint, thereby contradicting (11).

(d) Departure from Γ_{Ψ} but not from Γ_{PSAPF}:

This implies that

$$U_r \in \left[0,\ \sum_{\ell=1}^{Q_{\Psi}(t_j)} \alpha_\ell^{\Psi}(t_{j+1}^-)\right), \quad \text{and} \quad U_r \in \left[\sum_{\ell=1}^{Q_{PSAPF}(t_j)} \alpha_\ell^{PSAPF}(t_{j+1}^-),\ 1\right), \tag{12}$$

where $t_{j+1} = T_r$, the r^{th} potential completion instant, $1 \leq r < j+1$. Hence,

$$\sum_{\ell=1}^{Q_{PSAPF}(t_j)} \alpha_\ell^{PSAPF}(t_{j+1}^-) < \sum_{\ell=1}^{Q_{\Psi}(t_j)} \alpha_\ell^{\Psi}(t_{j+1}^-), \tag{13}$$

Using (10) we also have

$$\sum_{\ell=1}^{Q_{PSAPF}(t_j)} \alpha_\ell^{PSAPF}(t_{j+1}^-) \geq \sum_{\ell=1}^{Q_{PSAPF}(t_j)} \alpha_\ell^{\Psi}(t_{j+1}^-). \tag{14}$$

(13) and (14) together imply that

$$\sum_{\ell=1}^{Q_{PSAPF}(t_j)} \alpha_\ell^{\Psi}(t_{j+1}^-) \leq \sum_{\ell=1}^{Q_{PSAPF}(t_j)} \alpha_\ell^{PSAPF}(t_{j+1}^-) < \sum_{\ell=1}^{Q_{\Psi}(t_j)} \alpha_\ell^{\Psi}(t_{j+1}^-), \tag{15}$$

which shows that $Q_{PSAPF}(t_j) < Q_\Psi(t_j)$. Hence,

$$Q_{PSAPF}(t_{j+1}) = Q_{PSAPF}(t_j) \leq Q_\Psi(t_j) - 1 = Q_\Psi(t_{j+1}).$$

From (12) and (15) it also follows that $\sigma_k^\Psi(t_j)$ departs Γ_Ψ for some $k > Q_{PSAPF}(t_j)$. Hence,

$$M_i^{PSAPF}(t_{j+1}) = M_i^{PSAPF}(t_j) \geq M_i^\Psi(t_j) = M_i^\Psi(t_{j+1}), \quad i = 1, \ldots, Q_{PSAPF}(t_j) = Q_{PSAPF}(t_{j+1}).$$

This completes the proof by induction. Thus we have shown for every sample path that $Q_{PSAPF}(t) \leq Q_\Psi(t)$, $\forall\, t \geq 0$. Hence for every sample path

$$\overline{Q}_{PSAPF} = \lim_{t\to\infty} \frac{1}{t} \int_0^t Q_{PSAPF}(s)\, ds \quad \leq \quad \lim_{t\to\infty} \frac{1}{t} \int_0^t Q_\Psi(s)\, ds = \overline{Q}_\Psi,$$

from which it follows by Little's Law [18] that $\overline{R}_{PSAPF} \leq \overline{R}_\Psi$ for every sample path. Now uncondition on $\{A_j, N_j, C_j, T_j, U_j\}_{j=1}^\infty$. ∎

The arguments in the above proof can easily be reversed to show that PLAPF has the worst performance among all policies in Π_0^C when assumptions A1 and A2 hold. We need to restrict our attention to Π_0^C instead of Π_0 because PLAPF is a processor conserving policy.

Theorem 4.2 *Under assumptions A1 and A2,*

$$\overline{R}_\Psi \leq \overline{R}_{PLAPF}, \quad \forall \Psi \in \Pi_0^C.$$

Remark: Even though PSAPF is optimal under the above assumptions on the workload, it may not be a desirable policy for other practical considerations. In particular, knowledge of this policy could influence user behavior and create complex effects in the workload.

4.2 Applications of PSAPF and PLAPF bounds under A1 and A2

In this section we obtain the range of policy performance in Π_0^C for specific distributions of available parallelism, by comparing the PSAPF and PLAPF bounds for these distributions. We then compare specific policies in Π_0^C against the optimal PSAPF policy and also against the general lower bound under A1 and A2. As mentioned in Section 2.4 we assume that jobs arrive according to a Poisson arrival process with rate λ and that mean job demand $\overline{D} = P$. The mean response time estimates for all parallel processor policies in this section are obtained using discrete event simulation.

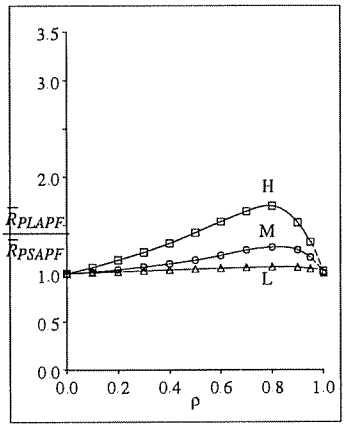
4.2.1 Range of Policy Performance in Π_0^C

Since PSAPF is optimal and PLAPF is pessimal in Π_0^C, under A1 and A2, the ratio of $\overline{R}_{PLAPF}$ to $\overline{R}_{PSAPF}$ measures the range of policy performance in Π_0^C. Figure 4 plots this ratio versus $\rho = \lambda\overline{D}/P$ for the H, M, and L workloads of Table 1 at P=20 and P=100. We note that the difference between PSAPF and PLAPF decreases as workload parallelism decreases. For $P = 20$ and $P = 100$ the difference between PSAPF and PLAPF for the L workload is very small which means that in this case any scheduling policy in Π_0^C performs nearly as well as the optimal PSAPF policy (when A1 and A2 hold). For the M and H workloads the difference between PSAPF and PLAPF is larger at P=100 than at P=20, indicating that for workloads with at least moderate parallelism the range of policy performance in Π_0^C increases with system size. (Note that for $P = 1$ all policies in Π_0^C have the same performance under A1 and A2.)

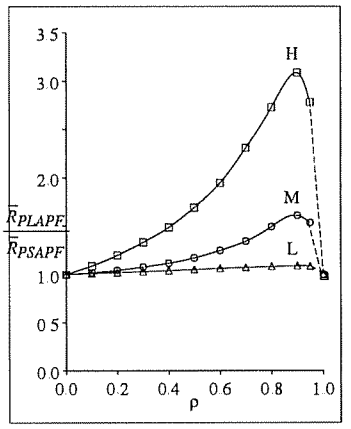
Another observation from these figures is that the ratio of PLAPF to PSAPF decreases as ρ goes to 1. (The dashed lines meeting at $\rho = 1$ are extrapolations from the simulation data points.) To explain this behavior we note that at very high utilizations there is a high probability of all P processors being occupied. Recall that under assumptions A1 and A2, departures occur with rate $P\mu$ for all policies in Π_0^C when all P processors are occupied, and thus as $\rho \rightarrow 1$ the system behavior approaches that of a saturated $G/M/1_P$ queue. Hence the performance of all policies in Π_0^C tends to converge as $\rho \rightarrow 1$.

4.2.2 Comparison of Specific Policies w.r.t. PSAPF and General Lower Bound

Figure 4 depicted the range of policy performance in Π_0^C for particular parallelism workloads, under assumptions A1 and A2. We now illustrate where specific policies in Π_0^C, namely FCFS and EQ, lie within this



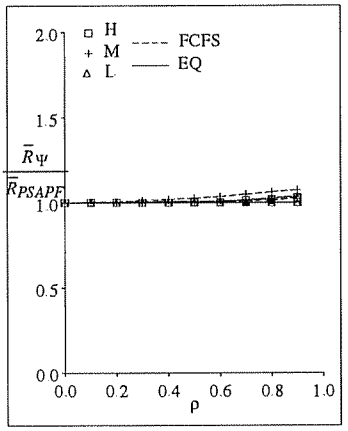
(a) P=20



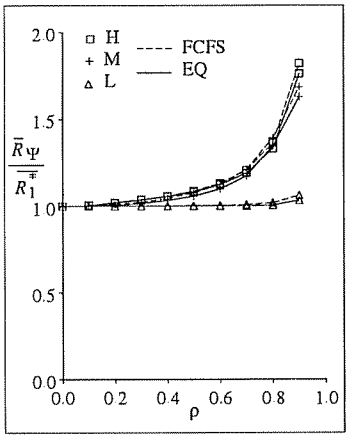
(b) P=100

Figure 4: Range of policy performance in Π_0^C under A1 and A2

range by comparing them with the PSAPF lower bound under A1 and A2, for the H, M, and L parallelism workloads. We then examine the performance of these two policies with respect to the general lower bound under the same workloads.



(a) $\overline{R}_{\Psi}/\overline{R}_{PSAPF}$



(b) $\overline{R}_{\Psi}/\overline{R}_1^*$

Figure 5: $\overline{R}_{\Psi}/\overline{R}_{PSAPF}$ and $\overline{R}_{\Psi}/\overline{R}_1^*$ under A1 and A2

$\Psi \in \{FCFS, EQ\}$, P=100

In Figure 5a we compare $\overline{R}_{FCFS}$ and $\overline{R}_{EQ}$ against the achievable PSAPF bound for the H,M, and L parallelism workloads. This figure clearly shows that $\overline{R}_{FCFS}$ and $\overline{R}_{EQ}$ are nearly identical to the optimal $\overline{R}_{PSAPF}$ for all three workloads at all utilizations, under A1 and A2. Figure 5b shows that if we instead

compare $\overline{R}_{EQ}$ and $\overline{R}_{FCFS}$ against the general lower bound, $\overline{R}_1^*$, we again see that FCFS and EQ have nearly identical performance for the H, M, and L workloads under A1 and A2, but we cannot determine how close these policies are to the optimal. Thus, knowledge of optimal policy performance (in Figure 5a) yielded useful information. Both Figures 5a and 5b show that $\overline{R}_{FCFS}$ is almost identical to $\overline{R}_{EQ}$ for the H,M, and L workloads, under A1 and A2. However, Figure 2 showed that $\overline{R}_{EQ}$ is significantly smaller than $\overline{R}_{FCFS}$ under the same parallelism workloads when D has an H_2 distribution with $C_D = 5$. Thus the qualitative behavior of processor allocation policies is sensitive to assumptions about job demand distribution.

4.3 Counterexamples

We have shown that PSAPF is the optimal policy in Π_0 and PLAPF the worst policy in Π_0^C under assumptions A1 and A2. The following counterexample shows that this is not the case for nonexponential demands (i.e., assumption A1 is no longer true). Note that Figure 2 showed that PSAPF is not optimal under specific parallelism workloads when job demand has an H_2 distribution with $C_D = 5$, but there was no mention about PLAPF in that counterexample. After the following counterexample we also give a counterexample to show that PSAPF is not optimal when A2 is violated.

Consider the following assumptions:

(a) $\{D_i\}_{i=1}^{\infty}$ are i.i.d. (nonexponential) and independent of everything else, with mean $\overline{D}$ and coefficient of variation C_D.

(b) $N \equiv P$, i.e., $N_i = P$, $i = 1, 2, \ldots$

(c) Linear job execution rates (assumption A2).

(d) Jobs arrive according to a Poisson process with rate λ.

Let $\rho = \lambda\overline{S} = \lambda\overline{D}/P$. Under assumption (b) there is just one priority class under PSAPF and PLAPF and therefore both policies reduce to the FCFS policy, for which the mean response time under assumptions (a)-(d) is the same as the mean response time of an $M/G/1$ queue with a processor of power P (i.e., $\overline{S} = \overline{D}/P$) [12]. Under assumptions (a)-(d) we therefore have

$$\overline{R}_{PSAPF} = \overline{R}_{PLAPF} = \overline{R}_{M/G/1_P} = \frac{\overline{D}}{P} + \frac{\rho(1+C_D^2)}{2(1-\rho)}\frac{\overline{D}}{P}, \tag{16}$$

where the expression for $\overline{R}_{M/G/1_P}$ is obtained from [7]. Now consider the EQ policy. The mean response time of the EQ policy under assumptions (a)-(d) is equal to the mean response time of an $M/G/1_P$ processor sharing (PS) queue [12]. From [8] we note that $\overline{R}_{M/G/1_P\ PS} = \overline{R}_{M/M/1_P}$. Thus,

$$\overline{R}_{EQ} = \overline{R}_{M/G/1_P\ PS} = \overline{R}_{M/M/1_P} = \frac{\overline{D}/P}{1-\rho}. \tag{17}$$

Comparing (16) with (17) shows that under assumptions (a)-(d) $\overline{R}_{PSAPF} > \overline{R}_{EQ}$ when $C_D > 1$, and $\overline{R}_{PLAPF} < \overline{R}_{EQ}$ when $C_D < 1$. Hence PSAPF is no longer optimal in Π_0 when job demand has a higher variance than the exponential distribution, and PLAPF no longer has the highest mean response time in Π_0^C when job demand has a smaller variance than the exponential. This counterexample exploited the first-come-first-serve ordering within PSAPF of jobs that have the same available parallelism. It can also be shown that for any other ordering of jobs with the same available parallelism (e.g., processor sharing) PSAPF is not optimal when A1 is violated.

The above counterexample assumed that A2 is true. When A2 is false it is easy to see that PSAPF is no longer optimal in Π_0 even when A1 is true. For example, if $N_i = P$ and $E_i(P) = c < P$, $i = 1, 2, \ldots$, then $\overline{R}_{PSAPF} = \infty$ for $\lambda > c/\overline{D}$. On the other hand, for the static policy in Π_0 that allocates exactly one processor to each job in FCFS order the stability condition is $\lambda < P/\overline{D}$, and thus its mean response time is smaller than $\overline{R}_{PSAPF}$ for $c/\overline{D} \leq \lambda < P/\overline{D}$.

5 Response Time Bounds for Exponential Demands: Optimal and Worst Case Parallelism

The bounds in the previous section were derived from the optimal policy in Π_0 and the pessimal policy in Π_0^C under assumptions A1 and A2, where assumption A1 states that $\{D_i\}_{i=1}^{\infty}$ are i.i.d. exponential with parameter μ and independent of everything else, and assumption A2 states that $E_i() = L_{N_i}()$, $i = 1, 2, \ldots$. In general, it is difficult to compute the mean response time of these policies (PSAPF and PLAPF), even under assumptions A1 and A2. In this section we derive computable, but weaker bounds on mean response time. In Section 5.1 we derive mean response time bounds in Π_0 and Π_0^C under assumptions A1 and A2 from optimal and worst case parallelism results. In Section 5.2 we show that these bounds can be computed for certain types of arrival processes. Section 5.3 tests the tightness of the these computable bounds against

the achievable PSAPF and PLAPF bounds under A1 and A2 and also compares the performance of specific policies under specific workloads against the mean response time under optimal parallelism. In Section 5.4 we give a counterexample to show that these bounds do not hold when assumption A1 is violated (i.e., jobs have nonexponential demands).

5.1 Parallelism Bounds Under A1 and A2

Under assumptions A1 and A2 we first derive a lower bound on mean response time for any policy $\Psi \in \Pi_0$ by showing that $\overline{R}_\Psi \geq \overline{R}_\Theta(N \equiv P)$, $\forall \Theta \in \Pi_0^C$. The notation $\overline{R}_\Theta(N \equiv P)$ stands for the mean response time of Θ when $N_i = P$, $i = 1, 2, \ldots$. We next derive an upper bound on mean response time for $\Theta \in \Pi_0^C$ by showing that $\overline{R}_\Theta \leq \overline{R}_\Theta(N \equiv 1)$, i.e., policy performance is worst when $N \equiv 1$.

The following lemma will be used in the bounds that follow.

Lemma 5.1 *Under assumptions A1 and A2*

$$\overline{R}_\Theta(N \equiv P) = \overline{R}_{G/M/1_P}, \quad \textit{and} \quad \overline{R}_\Theta(N \equiv 1) = \overline{R}_{G/M/P}, \quad \forall \Theta \in \Pi_0^C.$$

Proof. Let Γ_Θ denote a system with policy $\Theta \in \Pi_0^C$. Under assumptions A1 and A2, when $N \equiv P$ jobs depart from Γ_Θ with rate $P\mu$ as long as there is at least one job in the system. Hence the number of jobs in the system and consequently the mean response time of Γ_Θ are stochastically the same as that of the $G/M/1_P$ queue. Likewise, when $N \equiv 1$ jobs depart Γ_Θ with rate $\min(Q\mu, P\mu)$ when there are Q jobs in the system which is how the $G/M/P$ queue behaves. ∎

Lemma 5.1 shows that $\overline{R}_\Theta(N \equiv P)$ and $\overline{R}_\Theta(N \equiv 1)$ do not depend on Θ for $\Theta \in \Pi_0^C$, just like $\overline{R}_{G/M/c}$ for a $G/M/c$ queue does not depend on the scheduling policy as long as it is work-conserving and does not make use of job demands. Therefore under assumptions A1 and A2, all policies in Π_0^C have the same mean response time when $N \equiv P$ and also the same mean response time when $N \equiv 1$.

We now present a lower bound for the mean response time of policies in Π_0 that is derived from the optimal parallelism value for Π_0^C under assumptions A1 and A2.

Theorem 5.1 *Under assumptions A1 and A2*

$$\overline{R}_\Psi \geq \max(\overline{S}, \overline{R}_\Theta(N \equiv P)) = \max(\overline{S}, \overline{R}_{G/M/1_P}), \quad \forall \Psi \in \Pi_0, \quad \forall \Theta \in \Pi_0^C.$$

Proof. From Theorem 3.1

$$\overline{R}_{\Psi} \geq \max\left\{\overline{S}, \inf_{\Psi^1 \in \Pi_0^1} \overline{R}_{\Psi^1}[F_i() = L_P()]\right\},$$

where Ψ^1 is a nonidling uniprocessor policy. Under assumption A1 and when $F_i() = L_P()$, $i = 1, 2, \ldots$, (i.e., execution rate function $F_i(x) = x$, $0 \leq x \leq P$), the system under Ψ^1 behaves like a $G/M/1_P$ queue, $\forall \Psi^1 \in \Pi_0^1$. Hence

$$\overline{R}_{\Psi} \geq \max(\overline{S}, \overline{R}_{G/M/1_P}).$$

The theorem follows since from Lemma 5.1 we have $\overline{R}_{G/M/1_P} = \overline{R}_{\Theta}(N \equiv P)$, $\Theta \in \Pi_0^C$. ∎

Theorem 5.1 can also be proved independently of Theorem 3.1 by first coupling sample paths between Γ_{Ψ} (a system having policy $\Psi \in \Pi_0$ and a workload with general N_i, $i = 1, 2, \ldots$) and Γ_{Θ} (a system having policy $\Theta \in \Pi_0^C$ and a workload with $N_i = P$, $i = 1, 2, \ldots$) so that $\{A_i, C_i, T_i\}_{i=1}^{\infty}$ are the same for both systems, where T_i are potential job completion instants at jumps of a Poisson process with rate $P\mu$. In system Γ_{Θ} a potential completion is an actual job completion if there is at least one job in the system because $N \equiv P$, but this is not necessarily the case in Γ_{Ψ}. As a result, it can be shown by induction over time that for every sample path, $Q_{\Theta}(t) \leq Q_{\Psi}(t)$, $\forall t$, from which Theorem 5.1 follows.

An implication of Theorem 5.1 is that under A1 and A2 the performance of any policy in Π_0^C is best when $N \equiv P$. We now show that the reverse is true when $N \equiv 1$ by means of Theorem 5.2 below. The proof of this theorem is more complex than the proof of Theorem 5.1. Readers who wish to skip this proof on first reading can skip to Section 5.2 without loss of continuity.

Theorem 5.2 *Under assumptions A1 and A2*

$$\overline{R}_{\Psi} \leq \overline{R}_{\Theta}(N \equiv 1) = \overline{R}_{G/M/P}, \quad \forall \Psi, \Theta \in \Pi_0^C.$$

Proof. Let Γ_{Ψ} be a system having policy $\Psi \in \Pi_0^C$ and a workload with no restrictions on $\{N_i\}_{i=1}^{\infty}$. Let Γ_{Θ} be a system having policy $\Theta \in \Pi_0^C$ and a workload with $N_i = 1$, $i = 1, 2, \ldots$. Sort the jobs in Γ_{Φ} at time t in any fixed order, $\Phi \in \{\Psi, \Theta\}$ (for instance, in increasing order of available parallelism as in the proof of Theorem 4.1). Using the same notation as the proof of Theorem 4.1 job $\sigma_k^{\Phi}(t)$ departs Γ_{Φ} with rate $\alpha_k^{\Phi}(t)P\mu$, where $\alpha_k^{\Phi}(t)P$ is allocation of processing power at time t to job $\sigma_k^{\Phi}(t)$, $k = 1, \ldots, Q_{\Phi}(t)$, $\Phi \in \{\Psi, \Theta\}$.

Coupling of Sample Paths in Γ_Ψ and Γ_Θ

Fix $\{A_j, C_j, T_j\}_{j=1}^{\infty}$ as the same in both Γ_Ψ and Γ_Θ where T_j are potential completion instants generated by a Poisson process with rate $P\mu$. Fix a sequence $\{N_j\}_{j=1}^{\infty}$ for Γ_Ψ. By assumption A2 this automatically determines $\{E_j()\}_{j=1}^{\infty}$ for Γ_Ψ. In Γ_Θ all jobs have an available parallelism of 1 and this also automatically fixes their execution rates. To generate actual completions from Γ_Ψ and Γ_Θ let $\{U_j\}_{j=1}^{\infty}$ be i.i.d. Uniform[0,1) random variables. At the j^{th} potential completion instant T_j there is a departure from Γ_Φ if

$$U_j \in \left[0, \sum_{\ell=1}^{Q_\Phi(T_j)} \alpha_\ell^\Phi(T_j^-)\right), \quad \Phi \in \{\Psi, \Theta\}.$$

Sample Path Analysis

Using the above coupling of sample paths we show by induction over time that for every pair of coupled sample paths, for all $t \geq 0$

$$Q_\Psi(t) \leq Q_\Theta(t). \tag{18}$$

We carry out the induction over arrival instants and potential completion instants since no jobs depart in between these event times. Let $\{t_i\}_{i=0}^{\infty}$ be the sequence of arrival and potential completion times arranged in increasing order. Let both Γ_Ψ and Γ_Θ start out with zero jobs each. Then clearly (18) is satisfied at $t = t_0$. Assume that (18) is true for all $t \leq t_j$. Since no jobs arrive or depart in (t_j, t_{j+1}) (18) is satisfied for all $t < t_{j+1}$. We now prove that (18) is true at $t = t_{j+1}$. Consider all possible events at time t_{j+1}.

1. Job Arrival:

   By the induction hypothesis it follows that

   $$Q_\Psi(t_{j+1}) = Q_\Psi(t_j) + 1 \leq Q_\Theta(t_j) + 1 = Q_\Theta(t_{j+1}).$$

2. Potential Completion:

   (a) No departure from each of Γ_Ψ and Γ_Θ:

   $$Q_\Psi(t_{j+1}) = Q_\Psi(t_j) \leq Q_\Theta(t_j) = Q_\Theta(t_{j+1}).$$

   (b) Departure from Γ_Ψ but not from Γ_Θ:

   $$Q_\Psi(t_{j+1}) = Q_\Psi(t_j) - 1 \leq Q_\Theta(t_j) - 1 = Q_\Theta(t_{j+1}) - 1 < Q_\Theta(t_{j+1}).$$

(c) Departure from each of Γ_Ψ and Γ_Θ:

$$Q_\Psi(t_{j+1}) = Q_\Psi(t_j) - 1 \leq Q_\Theta(t_j) - 1 = Q_\Theta(t_{j+1}).$$

(d) Departure from Γ_Θ but not from Γ_Ψ:

This implies that

$$U_r \in \left[0, \sum_{\ell=1}^{Q_\Theta(t_j)} \alpha_\ell^\Theta(t_{j+1}^-)\right), \quad \text{and} \quad U_r \in \left[\sum_{\ell=1}^{Q_\Psi(t_j)} \alpha_\ell^\Psi(t_{j+1}^-),\ 1\right), \tag{19}$$

where $t_{j+1} = T_r$, the r^{th} potential completion instant, $1 \leq r < j+1$. Hence,

$$\sum_{\ell=1}^{Q_\Psi(t_j)} \alpha_\ell^\Psi(t_{j+1}^-) < \sum_{\ell=1}^{Q_\Theta(t_j)} \alpha_\ell^\Theta(t_{j+1}^-). \tag{20}$$

From (19) it follows that there are idle processors in Γ_Ψ at time t_{j+1}^-. Thus at t_{j+1}^- job $\sigma_k^\Psi(t_{j+1}^-)$ has as many processors as its available parallelism, $k = 1, \ldots, Q_\Psi(t_j)$, because Ψ is processor conserving. As a result,

$$\alpha_\ell^\Psi(t_{j+1}^-) \geq \frac{1}{P} \geq \alpha_\ell^\Theta(t_{j+1}^-), \quad \ell = 1, \ldots, Q_\Psi(t_j). \tag{21}$$

(20) and (21) together imply

$$\sum_{\ell=1}^{Q_\Psi(t_j)} \alpha_\ell^\Theta(t_{j+1}^-) \leq \sum_{\ell=1}^{Q_\Psi(t_j)} \alpha_\ell^\Psi(t_{j+1}^-) < \sum_{\ell=1}^{Q_\Theta(t_j)} \alpha_\ell^\Theta(t_{j+1}^-),$$

which shows that $Q_\Psi(t_j) < Q_\Theta(t_j)$. Hence

$$Q_\Psi(t_{j+1}) = Q_\Psi(t_j) \leq Q_\Theta(t_j) - 1 = Q_\Theta(t_{j+1}).$$

This completes the proof by induction. Thus we have shown for every sample path that $Q_\Psi(t) \leq Q_\Theta(t)$, $\forall\, t \geq 0$. Hence for every sample path

$$\overline{Q}_\Psi = \lim_{t\to\infty} \frac{1}{t} \int_0^t Q_\Psi(s)\, ds \quad \leq \quad \lim_{t\to\infty} \frac{1}{t} \int_0^t Q_\Theta(s)\, ds = \overline{Q}_\Theta,$$

from which it follows by Little's Law that $\overline{R}_\Psi \leq \overline{R}_\Theta$ for every sample path. Now uncondition on $\{A_j, N_j, C_j, T_j, U_j\}_{j=1}^\infty$. From Lemma 5.1 we also know that $\overline{R}_\Theta = \overline{R}_{G/M/P}$ since $N \equiv 1$ in Γ_Θ. ∎

5.2 Computation of Parallelism Bounds

From the two theorems of this section and from the theorems of Section 4 we have the following corollary.

Corollary 5.1 *Under assumptions A1 and A2*

$$\overline{R}_{\Psi} \geq \overline{R}_{PSAPF} \geq \max(\overline{S}, \overline{R}_{G/M/1_P}), \quad \forall \Psi \in \Pi_0 \quad \textit{and}$$

$$\overline{R}_{\Theta} \leq \overline{R}_{PLAPF} \leq \overline{R}_{G/M/P}, \quad \forall \Theta \in \Pi_0^C.$$

We refer to the lower bound $max(\overline{S}, \overline{R}_{G/M/1_P})$ as the $N \equiv P$ bound and the upper bound $\overline{R}_{G/M/P}$ as the $N \equiv 1$ bound. The $N \equiv P$ and $N \equiv 1$ bounds are looser than the $\overline{R}_{PSAPF}$ and $\overline{R}_{PLAPF}$ bounds, but they are useful because they can be computed exactly for certain arrival processes, which is not necessarily the case for $\overline{R}_{PSAPF}$ and $\overline{R}_{PLAPF}$. For example, for a Poisson arrival process the bounds in Theorems 5.1 and 5.2 reduce to

$$\overline{R}_{\Psi} \geq \max\left(\overline{S}, \frac{\overline{D}/P}{1-\rho}\right), \quad \forall \Psi \in \Pi_0, \qquad \text{and} \qquad \overline{R}_{\Theta} \leq \overline{R}_{M/M/P}, \quad \forall \Theta \in \Pi_0^C. \tag{22}$$

The second term within the max() in the lower bound is simply the mean response time of an $M/M/1_P$ queue with mean service time $\overline{D}/P$. When the arrival process is GI but not necessarily Poisson the parallelism bounds can be computed using the analysis of the $GI/M/c$ queue [7,21].

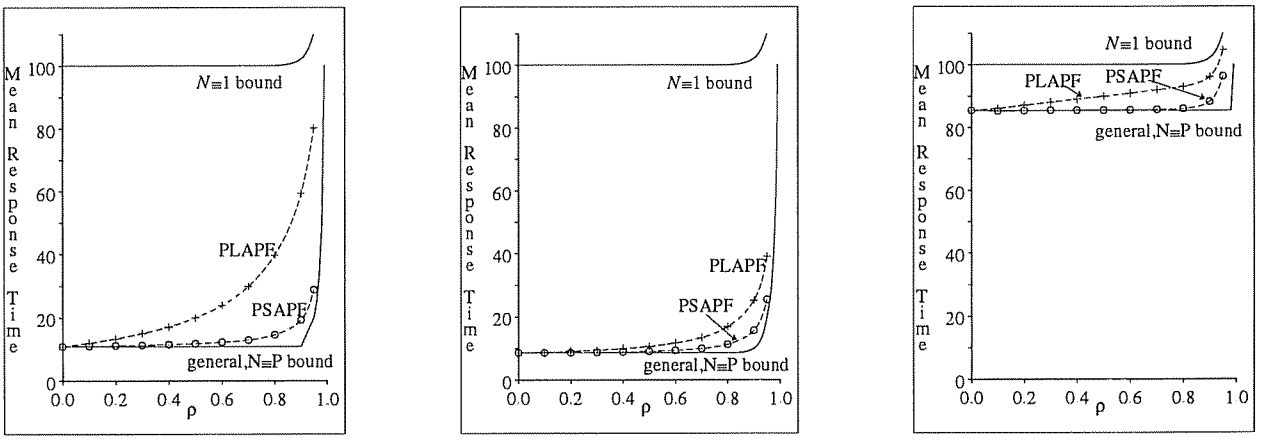
5.3 Experimental Results

In this section we first compare the computable parallelism bounds ($N \equiv P$ and $N \equiv 1$ bounds) against the achievable PSAPF and PLAPF bounds, for specific distributions of N. We then illustrate the use of the $\overline{R}_{\Psi}(N \equiv P)$ ($\Psi \in \Pi_0^C$) result by comparing the performance of the PSAPF, FCFS and EQ policies for specific distributions of N against the performance of the same policies under fully parallel workloads. As before we assume that jobs arrive according to a Poisson process with rate λ and that $\overline{D} = P$.

5.3.1 Comparison of Computable versus Achievable Bounds

Recall that under assumptions A1 and A2 the general lower bound is equivalent to the $N \equiv P$ bound and we therefore refer to it as the "general,N≡P" bound. For P=100, Figure 6 presents plots of the general,N≡P bound and the $N \equiv 1$ bound as well as the PSAPF and PLAPF bounds for the H, M, and L workloads.

The results for P=20 (not shown) are qualitatively the same. The difference between $\overline{R}_{PLAPF}$ and $\overline{R}_{PSAPF}$ gives another view of the range of policy performance in Π_0^C that was discussed in Section 4.2.1. Comparing the computable $N \equiv 1$ upper bound versus the achievable PLAPF upper bound for Π_0^C, we observe that among the H, M, and L workloads the $N \equiv 1$ bound is tightest for the L workload (as expected). However, the $N \equiv 1$ bound is generally loose with respect to $\overline{R}_{PLAPF}$ for all three parallelism workloads except at very high utilizations.



(a) H workload (b) M workload (c) L workload

Figure 6: Response time bounds under A1 and A2

P=100

One implication of these results is that under assumptions A1 and A2 the performance of any policy in Π_0^C is much better under parallel workloads than under fully sequential workloads. However, in Section 5.4 will show that this observation can be sensitive to assumptions A1 and A2 for specific policies in Π_0^C.

Comparing the computable general,N≡P lower bound versus the achievable PSAPF lower bound, we observe from Figure 6 that for the H and M workloads the general,N≡P bound is reasonably tight, particularly at low to moderate utilizations as well as at very high utilizations ($\rho \geq 0.9$), and the location of the knee of the PSAPF curve is predicted quite well by the general,N≡P curve. For the L workload, the general,N≡P bound is tight for most of the range of utilization, but is loose at the knee of the PSAPF curve. Based on the results of Section 4.2.2 shown in Figure 5a, the FCFS and EQ curves would lie very close to the PSAPF curves in Figure 6.

5.3.2 Comparison of PSAPF, FCFS, and EQ against $\overline{R}_{\Psi}(N \equiv P)$

We compare the mean response time of particular policies in Π_0^C under specific distributions for N against the mean response time when $N \equiv P$, to get an estimate of the benefits of full parallelism in the workload. We consider the PSAPF, FCFS, and EQ policies which were shown in Figure 5a to have almost the same performance for the H, M, and L workloads under assumptions A1 and A2. Figure 7 plots the ratio of $\overline{R}_{EQ}$ to $\overline{R}_{EQ}(N \equiv P)$, under assumptions A1 and A2, for the H, M, and L workloads; P=100. The curves for PSAPF and FCFS are almost identical to the curves for EQ. Since $\overline{D} = P$ for this experiment $\overline{R}_{EQ}(N \equiv P) = \overline{D}/P = 1$ at $\rho = 0$, and thus the ratio $\overline{R}_{EQ}/\overline{R}_{EQ}(N \equiv P) = \overline{S}/1 = \overline{S}$ at $\rho = 0$. For the workloads in Table 1, $\overline{S}$ is lower for the M workload than for the H workload. Note, however, that there are many workloads with moderate $\overline{N}$ (not shown) that have higher mean service time than the H workload. Each of the curves in Figure 7 starts at $\overline{S}$ at $\rho = 0$ and decreases linearly with ρ until it reaches the limiting value of 1.0 at $\rho = 1$. (The dashed lines leading to 1.0 at $\rho = 1$ are extrapolations from the simulation data points.) The reason for the decrease is that an increase in processor utilization causes departures to occur at higher rates under assumptions A1 and A2, until they reach the limiting rate of $P\mu$ at $\rho = 1$. Though all curves in Figure 7 converge at $\rho = 1$, the figure shows that a substantial improvement in mean response time is possible, even at high utilizations such as $\rho = 0.9$, by making each of the workloads fully parallel; this is particularly true for the L workload.

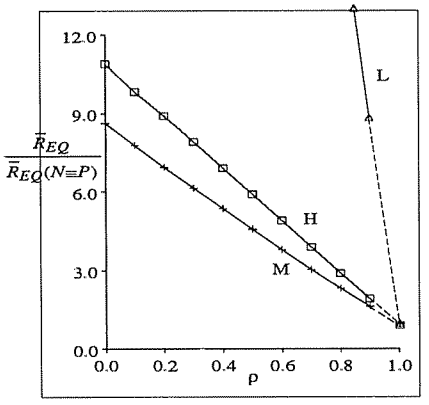


Figure 7: $\overline{R}_{EQ}/\overline{R}_{EQ}(N \equiv P)$ under A1 and A2

P=100

The above results are derived under assumptions A1 and A2. We next discuss the sensitivity of the parallelism bounds to these assumptions.

5.4 Counterexample for nonexponential job demands

We have shown that

$$\overline{R}_\Theta(N \equiv P) \leq \overline{R}_\Theta \leq \overline{R}_\Theta(N \equiv 1), \quad \forall \Theta \in \Pi_0^C, \tag{23}$$

when jobs have i.i.d. exponential demands independent of everything else, and execution rates are linear. It is easy to see that these bounds are violated for some policies in Π_0^C when assumption A2 is false. For example, for the FCFS policy with $N \equiv P$ and $E_i(P) < P$, for all i, the system will become unstable at lower arrival rates than when $N \equiv 1$ (under which execution rates are linear). We now give a counterexample to show that (23) does not hold for all policies in Π_0^C when assumption A1 is false. Consider the following assumptions:

(a) $\{D_i\}_{i=1}^{\infty}$ are i.i.d. with distribution B (given below) and independent of everything else. The distribution B is a generalized exponential distribution (see [1]) defined by

$$B(t) = \begin{cases} 0, & t < 0, \\ 1 - \beta e^{-\alpha t}, & t \geq 0, \end{cases}$$

where $0 < \beta \leq 1$ and $\alpha > 0$.

(b) Linear job execution rates (assumption A2)

(c) Jobs arrive according to a Poisson process with rate λ.

Denote the mean of the distribution B by $\overline{D}$ and let $\rho = \lambda \overline{D}/P$. Consider the FCFS policy in Π_0^C. Under assumptions (b) and (c) when $N \equiv P$ the mean response time under FCFS is that of an $M/G/1_P$ queue and when $N \equiv 1$ the mean response time under FCFS is that of an $M/G/P$ queue [12]. Under assumptions (a)-(c) and for $P = 2$ it follows (see [1]) that

$$\overline{R}_{FCFS}(N \equiv P = 2) = \overline{R}_{M/G/1_2} > \overline{R}_{M/G/2} = \overline{R}_{FCFS}(N \equiv 1), \quad \text{iff } \rho > \frac{\beta}{1-\beta},$$

which violates (23) when $\beta < 1/2$. Hence the parallelism bounds of this section do not hold for all policies in Π_0^C under nonexponential job demands. (They may, however, continue to hold for some policies in Π_0^C

even under nonexponential demands.) This again shows that the qualitative behavior of scheduling policies is sensitive to the assumptions about job demand distribution.

6 Conclusions

We have developed a general lower bound for the mean response time of a general purpose parallel processor system with a central job queue. The lower bound for parallel processor policies is derived for a general workload model from the optimal nonidling uniprocessor policy that uses the same information as the parallel processor policy. We have also given examples of how tighter bounds can be obtained on a per policy basis. Key features of the workload model are general distributions of demand and available parallelism, general nonlinear job execution rates, and general inter-arrival times, with arbitrary dependencies between these variables.

Under a restrictive set of assumptions that includes exponential i.i.d. job demands independent of all other workload variables, and linear job execution rates, we have shown that when job demand is not explicitly known to the scheduler the PSAPF policy is optimal and the PLAPF policy is pessimal over all *processor conserving* policies. We have also given a counterexample to show that PSAPF optimality does not hold for distributions with higher variance than the exponential. Under the same set of assumptions, we have shown that the mean response time of any processor conserving policy is lowest when every job can make use of all processors and is highest when all jobs are fully sequential. We have given a counterexample to show that these parallelism bounds are violated when job demand is not exponential.

We have given some experimental data to illustrate the applicability and tightness of the derived bounds under specific demand and parallelism workloads. These experimental results show that the general lower bound serves as a useful reference point to compare the performance of parallel processor policies in a given class, especially when the optimal parallel processor policy in the same class is unknown for the given workload assumptions. The data also indicates that per-policy bounds can be considerably tighter than the general lower bound for certain workloads, but they may not be much tighter for workloads that have a substantial fraction of jobs with very low parallelism. Under exponential job demands and linear execution rates the data provided examples of processor conserving policies that perform nearly as well as the optimal processor conserving policy for specific workloads of available parallelism. Under the same assumptions the data have also shown that substantial improvements in mean response time are possible by making specific

workloads fully parallel. Note that this last result will hold for specific policies under more general demands and linear execution rates, if the performance of such policies is insensitive to demand distribution.

References

1. BRUMELLE, S. Some inequalities for parallel-server queues. *Oper. Res. 19*, 2 (1971), 402-413.

2. CONWAY, R., MAXWELL L., AND MILLER, L. *Theory of Scheduling.* Addison-Wesley, Reading, Massachusetts, 1967.

3. FAYOLLE, G., MITRANI, I., AND IASNOGORODSKI, R. Sharing a processor among many job classes. *J. ACM 27*, 3 (Jul. 1980), 519-532.

4. FERGUSON, M. Weighted processor sharing - results for hyperexponential servers. *IEEE Trans. on Software Engg., SE-9*, 4 (Jul. 1983), 531-535.

5. HIRAYAMA, T., AND KIJIMA, M. An extremal property of FIFO discipline in G/IFR/1 queues. *Adv. Appl. Proba. 21*, 2 (1989), 481-484.

6. KLEINROCK, L. Time-shared systems: a theoretical treatment. *J. ACM 14*, 2 (Apr. 1967), 242-261.

7. KLEINROCK, L. *Queueing Systems, Vol I: Theory.* John Wiley & Sons, New York 1975.

8. KLEINROCK, L. *Queueing Systems, Vol II: Computer Applications.* John Wiley & Sons, New York 1976.

9. LEUTENEGGER, S., AND NELSON, R. Analysis of spatial and temporal scheduling policies for semi-static and dynamic multiprocessor environments. Research Report, IBM T.J. Watson Research Center, Yorktown Heights, Aug. 1991.

10. LEUTENEGGER, S., AND VERNON, M. The performance of multiprogrammed multiprocessor scheduling policies. *Proc. ACM SIGMETRICS Conf. on Measurement & Modeling of Computer Systems 18*, 1 (May 1990), 226-236.

11. MAJUMDAR, S., EAGER, D., AND BUNT, R. Scheduling in multiprogrammed parallel systems. *Proc. ACM SIGMETRICS Conf. on Measurement & Modeling of Computer Systems 16*, 1 (May 1988), 104-113.

12. MANSHARAMANI, R., AND VERNON, M. Approximate analysis of parallel processor allocation policies. *In preparation.*

13. SCHRAGE, L. The queue M/G/1 with the shortest remaining processing time discipline. *Oper. Res. 14*, 4 (1966), 670-684.

14. SCHRAGE, L. A proof of the optimality of the shortest remaining processing time discipline. *Oper. Res. 16*, 3 (1968), 687-690.

15. SCHRAGE, L. Optimal scheduling disciplines for a single machine under various degrees of information. Grad. School of Business, Univ. of Chicago, 1974.

16. SETHI, R. On the complexity of mean flow time scheduling. *Math. of Oper. Res. 2*, 4 (1977), 320-330.

17. SEVCIK, K. Application scheduling and processor allocation in multiprogrammed parallel processing systems. To appear in a special issue of *Performance Evaluation.*

18. STIDHAM, S. A last word on $L = \lambda W$. *Oper. Res. 22*, 2 (1974), 417-421.

19. THAKKAR, S., GIFFORD, P., AND FIELLAND, G. Balance: a shared memory multiprocessor system. *Proc. Int. Conf. on Supercomputing 2*, (1987), 93-101.

20. WALRAND, J. *Introduction to Queueing Networks*. Prentice-Hall, Englewood Cliffs, New Jersey, 1988.

21. WOLFF, R. *Stochastic Modeling and the Theory of Queues*. Prentice-Hall, Englewood Cliffs, New Jersey, 1989.