Distributed Parallelism Considered Harmful

by A.M. Makowski and R. Nelson
Distributed Parallelism Considered Harmful

Armand M. Makowski
Electrical Engineering Department
and Systems Research Center
University of Maryland
College Park, MD 20742

Randolph Nelson
IBM Research Division
T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

December 4, 1991

Abstract

We consider a model of a distributed parallel processing system that shows that parallel versus sequential processing is beneficial only under conditions of light load. Our results are valid under general assumptions on the number of processors, task service times and the information used to schedule jobs. Our model of a parallel processing system consists of a set of homogeneous processors each with private memory in which tasks queue before being served. Jobs arriving to the system consist of a random number of tasks which can be executed independently of each other and we consider a job to be completed only after all of its component tasks have finished execution. A central dispatcher schedules the tasks on the processors at job arrival instants using information on the number of tasks currently scheduled on each processor. We model this system as a distributed fork/join queueing system and derive the structure of the individually optimal scheduling policy. Our results show that the individually optimal policy is a mixture of policies corresponding to sequential job execution (all tasks are scheduled on a single processor) and parallel scheduling (tasks are distributed among several processors in a manner that tends to equalize queue lengths). We show that, under conditions that include the case of moderate to heavy loads, the individually optimal scheduler schedules tasks according to the sequential policy which runs counter to the intuition that parallel processing is desirable. Because we do not include certain overheads associated with executing jobs in parallel in our model, our results are biased towards parallel rather than sequential processing. Since we believe that systems are not typically underutilized, our results strongly suggest that it can be harmful to have parallel execution in distributed processing systems. Response time properties of the individually optimal scheduler are derived and compared to other scheduling policies.
1 Introduction

Parallel processing offers the advantages of speeding up job execution and of balancing system load across a set of multiple processors. The costs incurred in executing jobs in parallel range from contention for shared resources such as memory access and interconnection bandwidth, to overheads associated with synchronizing the computation among the tasks [10, 16, 30]. Synchronization costs include the time and resources used in the synchronization protocol to establish when all tasks have finished execution as well as inherent randomness associated with task execution times. The underlying processor architecture clearly has an effect on these costs as do characteristics of the policy used to schedule jobs. Previous work on these issues has been carried out in both deterministic and probabilistic frameworks. Deterministic models that assume unbounded parallelism and include effects of the synchronization protocols are discussed in [1, 9, 10, 15, 16]; models that ignore these costs and analyze the effects that randomness in task execution time have on synchronization are considered in [2, 3, 4, 5, 31, 32]. Work relating the effects of scheduling policies on response times on a parallel processing environment that include synchronization overheads can be found in [22, 25, 36, 40].

In this paper we consider the effects that the scheduling policy has on the expected job response time when the synchronization costs arise among other things from the randomness associated with task execution times. The parallel processing system we consider here consists of a set of distributed homogeneous processors each with a local memory from which it selects tasks to serve. We assume that jobs arrive to a central queue from which they are dispatched among the processors. Once a task is scheduled on a given processor we assume that it will be executed on that processor. That is, we concern ourselves with static scheduling policies, in contrast which dynamic policies which allow tasks to migrate between processors as a function of the state of the system. We point the reader to references [11, 12, 13, 20, 21, 24, 28, 29, 38] where migration policies for sequential jobs are analyzed under various system architectures. Our model is applicable to situations where such migrations are not allowed. Such a restriction might be imposed on the scheduler to limit scheduling complexity or because the benefits of a dynamic policy are determined to be small with respect to the costs associated with migrating tasks. The results we present here can, we believe, be used to guide the construction of dynamic policies for distributed systems. For example, it might be reasonable to initially schedule
tasks in the system according to the static policy under the assumption that there is a large probability that no migrations take place before a scheduled job completes execution. Examples of applications of our model are that of a set of independent processors that communicate over a local area network. An example of such a system is the Condor system at the University of Wisconsin which provides facilities for distributed processing over a set of workstations [23]. Another situation that can be modeled can be found in computer systems with non-uniform memory access (NUMA) where the cost to access nonlocal memory is very expensive.

In our model, jobs in the system are assumed to consist of a number of tasks which can be independently scheduled on the processors of the system; we assume that a job finishes execution only after all of its tasks finish executing. In [17, 32] a similar model was analyzed under the assumption that all tasks of a job were evenly distributed among the processors of the system, while in [33, 34] all tasks were assumed scheduled on the processor that had a minimal number of tasks in its local queue. The relative performance of these two scheduling policies depends on system utilization, i.e., scheduling all the queues is best at low utilizations whereas scheduling only the shortest queue is best at high utilizations.

In this paper and in the companion paper [26] which deals with a more general case, we identify the policy which minimizes the expected execution time of each individual job. This policy, termed the individually optimal policy, differs from a socially optimal policy which attempts to minimize the expected job response time for all job arrivals to the system. The individually optimal policy can be derived in first-come first-serve systems by not accounting for the effects that scheduling decisions have on future arrivals. Although future arrivals do influence the expected response time averaged over all arrivals, we believe that the policy derived here is sufficiently close to the socially optimal policy to be of practical use.

Perhaps more importantly, the analysis reveals some interesting tradeoffs found in parallel processing scheduling. Some of our results are surprising in that they contradict the intuition that parallel processing is beneficial. In fact, we show that naively using parallel processing can be harmful to the system. As an overall observation, we find that the optimal policy tends to use parallel processing only for low to medium utilisations where the benefits of distributing the load across several processors outweighs the costs associated with synchronizing random task execution times. This result is strengthened by the fact that our model ignores overheads associated with executing jobs in parallel and thus is biased towards parallel rather than sequential
processing. Additionally, since we believe that, for obvious cost reasons, most systems are run under conditions of high utilization, executing jobs in parallel can actually be harmful to the system in that it increases mean response time and decreases maximum throughput (since there are overheads associated with parallel execution). These results are robust to the number of processors, distribution of task size and information used by the system to schedule jobs. A detailed analysis of this phenomena is found in our results.

In this paper we derive the structural properties of the optimal policy only for the case of two processors. Additionally we restrict task distributions to be exponential. Restricting our attention to this simpler case allows for a thorough self-contained discussion with a minimum of formalism while still retaining the salient features of the problem. Moreover, the developments below establish a basis for the general case which is solved in [26]. However, for an arbitrary number of processors, and general task distributions the analysis is far more delicate as it combines ideas from the theory of stochastic orderings [37] with the notion of majorization [27]. We emphasize here that although we only consider a special case, our conclusions are based on the fact that our results hold generally.

We have organized the discussion as follows: Section 2 presents the mathematical model that we use to establish properties of the optimal scheduler. In section 3 we derive the load balancing properties of the optimal scheduler, while in section 4 we show that sequential processing is eventually preferred over parallel processing in “high” loads. In section 5 we discuss our results on the expected response time for the optimal policy, and we compare them to the two policies mentioned above, namely the one which schedules all the queues evenly and the one which only schedules the shortest queue. In section 6 we present our conclusions and suggest future research. The appendices contain several technical results which are not essential for understanding the main conclusions of the paper.

A few words on the notation used in this paper: We denote the set of non-negative integers by \( \mathbb{N} \), and the set of all real numbers by \( \mathbb{R} \). For every positive integer \( L \), the \( l^{th} \) component of any element \( x \) in \( \mathbb{R}^L \) is denoted by \( x_l \), \( l = 1, \ldots, L \), so that \( x \equiv (x_1, \ldots, x_L) \).
2 The Model And The Scheduling Problem

Here, we consider the queueing model, depicted in figure 1, consisting of $K \geq 2$ homogeneous, single server queues. Each queue has infinite capacity. Task service times at the queues are independent and identically distributed (i.i.d.) non-negative random variables (r.v.'s). We let $X$ be the r.v. denoting the generic service time at any one of the queues and we denote its expected value by $\bar{X}$ and its variance by $\sigma_X^2$. Each incoming job is assumed to consist of a random number of tasks described by an integer-valued r.v. $B$. A job is considered to be finished when all its tasks have finished execution. This corresponds to a synchronization event in the underlying parallel processing model.

Although the model description in this paper is given for general $K$, we derive here the optimal policy and its properties only for the $K = 2$ case. This allows us to establish certain structural properties of the optimal policy found in this case using a minimal of formalism. These properties shed light on some non-obvious trade-offs found between parallel and sequential processing and provide the first hints of the structure of the optimal policy for the general case. As mentioned in the introduction, the derivation of the optimal policy for general $K$ is substantially more complex and requires formalism's that lie outside the scope of this paper. The interested reader can find this derivation in the companion paper [26].

We denote by $n_k$ the number of tasks in queue $k, k = 1, 2, \ldots, K$, at a job arrival instant and we write $n \equiv (n_1, n_2, \ldots, n_K)$ for the vector state of the system (which is an element of $\mathbb{N}^K$). Upon job arrival, the value $b$ of $B$ is declared and a dispatcher schedules the job's tasks on the queues using the values of $n$ and of $B$. We are interested in determining the scheduling policy which, for each arriving job, schedules tasks so that the expected response time of the job is minimized. This policy is the individually optimal policy for the system, and as noted in the introduction, differs from the socially optimal policy which minimizes the expected response time over all job arrivals. To avoid cumbersome language, henceforth when we say optimal policy, we mean individually optimal policy.

To define the optimal scheduling problem considered in this paper, let $\{X^h_j, j = 1, 2, \ldots, k = \ldots, K\}$. 

4
1, 2, . . . , K) be i.i.d. non-negative r.v.'s distributed as X and set

\[ S_n^k = \sum_{j=1}^{n} X_j, \quad k = 1, 2, \ldots, K, \quad n = 1, 2, \ldots \]  

(1)

with the convention \( S_0^k = 0 \). The value of \( S_n^k \) is the time needed to execute all the jobs scheduled on processor \( k \) given that its queue contains \( n \) tasks. Note that \( ES_n^k = n\bar{X} \). Our assumptions imply that scheduling decisions are made using information that can be summarized by the number of tasks already on processor queues. In particular, we assume either that tasks in the servers have just started execution upon job arrival or that the remaining service time (i.e., residual life) has an identical distribution to that of a service time (in which case the service times are necessarily exponentially distributed).

An admissible scheduling policy is a mapping

\[ \pi : \mathbb{N} \times \mathbb{N}^K \rightarrow \mathbb{N}^K : (b, n) \rightarrow \pi(b; n) \equiv (\pi_1(b; n), \ldots, \pi_K(b; n)) \]  

(2)

satisfying

\[ \sum_{k=1}^{K} \pi_k(b; n) = b, \quad b = 1, 2, \ldots, \quad n \in \mathbb{N}^K. \]  

(3)

In words, when the system is in state \( n \) and the incoming job is made up of \( b \) tasks, policy \( \pi \) schedules \( \pi_k(b; n) \) of the incoming tasks on processor \( k, k = 1, 2, \ldots, K \). The resulting response time of this incoming job is given by

\[ T_\pi(b; n) = \max_{k: \pi_k(b; n) > 0} \left\{ S_{n+\pi_k(b; n)}^k \right\}, \quad b = 1, 2, \ldots, \quad n \in \mathbb{N}^K. \]  

(4)

We let

\[ \bar{T}_\pi(b; n) = ET_\pi(b; n), \quad b = 1, 2, \ldots, \quad n \in \mathbb{N}^K \]  

(5)

be the expected response time of the incoming job of size \( b \) when the state of the system at time of arrival is \( n \) and policy \( \pi \) is used.

The collection of all admissible scheduling policies is denoted by \( \mathcal{P} \). We seek to determine the scheduling policies in \( \mathcal{P} \) which minimizes the expected response time \( (5) \). We denote any such optimal policy by \( \Psi \) which is characterized by

\[ \bar{T}_\Psi(b; n) \leq \bar{T}_\pi(b; n), \quad \pi \in \mathcal{P}, \quad b = 1, 2, \ldots, \quad n \in \mathbb{N}^K. \]  

(6)
In the remainder of the paper, we consider the case $K = 2$ so that the state upon the arrival of a tagged job is given by $n = (n_1, n_2)$ and there are $b \geq 1$ tasks to be scheduled. In discussing the properties of the optimal schedule $\Psi$, it is clear that we can relabel the queues without loss of generality in our results. This is a direct consequence of the underlying statistical assumptions and of the symmetry property of the maximum operator. Therefore, to simplify the notation at various places in the exposition, it will be convenient to assume a labeling of the queues so that $n_1 \leq n_2$. Thus we can consider queue 1 (resp. 2) to be the shortest (resp. longest) queue (with the convention that queue 1 is shortest if $n_1 = n_2$). We also define $\Delta n \equiv n_2 - n_1$.

3 Load Balancing Properties of The Optimal Policy

The operation of the optimal policy can be considered to proceed in two steps. First the policy must determine whether one or both queues are to be scheduled. If only one queue is to be scheduled, then the policy must determine which queue is allocated the $b$ tasks. If both queues are to be scheduled, it then suffices to determine how many tasks are assigned to each queue. This two step procedure is formalized in Lemma 1 and Theorem 2 below which contain the first key structural properties of the optimal policy $\Psi$.

To facilitate the presentation we will say that if, for the given state $n$, the policy $\Psi$ schedules the $b$ tasks on more than one queue, then $BQ$ (both queues) is satisfied in state $n$, otherwise if $\Psi$ schedules all $b$ tasks on a single queue, then $SQ$ (single queue) is satisfied in state $n$. The collection of all non-degenerate allocation vectors for assigning $b$ tasks to both queues is the set $B(b)$ given by

$$B(b) \equiv \{b = (b_1, b_2) \in \mathbb{N}^2 : 0 < b_1, b_2 < b; \ b_1 + b_2 = b\}. \quad (7)$$

Using the definitions, we see that $BQ$ (resp. $SQ$) is satisfied in state $n$ when scheduling $b$ tasks provided the condition

$$\min_{b \in B(b)} E \max_{k=1,2} \{S_{n_k+b_k}^b\} \leq (\text{resp. } \geq) \min_{k=1,2} ES_{n_k+b}^b \quad (8)$$

holds with the convention that the minimum over an empty set is $+\infty$.

In this section we establish load balancing properties of the optimal policy. We start by identifying obvious conditions where $SQ$ is satisfied.
Lemma 1. Suppose that in state $n$, the optimal policy $\Psi$ schedules $b \geq 1$ tasks. Then $SQ$ is necessarily satisfied if either $b = 1$ or $1 \leq b \leq \Delta n$.

Proof. The case $b = 1$ is a trivial consequence of (8) since $B(b)$ is then empty. Next assume $1 < b \leq \Delta n$, and for every allocation vector $b$ in $B(b)$, we thus have

$$ES_{n_1+b}^1 = (n_1 + b)\bar{X} \leq n_2\bar{X} < (n_2 + b_2)\bar{X} = ES_{n_1+b_2}^2$$

so that

$$ES_{n_1+b}^1 < \max_{k=1,2} S_{n_k+b_k}^k.$$ 

This establishes the result. 

We note from the strict inequality in (10) that $BQ$ cannot hold in state $n$ when $1 \leq b \leq \Delta n$.

We now state a result that shows if the optimal policy $\Psi$ schedules on both queues, it does so in a way that tends to equalize their queue lengths. In what follows, $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ denote the floor and ceiling functions, respectively ([19], p. 37).

Theorem 2. Suppose that in state $n$, the optimal policy $\Psi$ schedules $b \geq 1$ tasks.

1. If $SQ$ is satisfied in state $n$, then $\Psi$ schedules all $b$ tasks on queue 1;

2. If $BQ$ is satisfied in state $n$, then necessarily $2 \leq b$ and $\Delta n < b$, and $\Psi$ schedules $l_k(b;n)$ tasks on queue $k$, $k = 1, 2$, with $l_k(b;n)$ given by

$$l_k(b;n) \equiv \begin{cases} \Delta n + \lfloor \frac{b-\Delta n}{2} \rfloor, & k = 1, \\ \lceil \frac{b-\Delta n}{2} \rceil, & k = 2. \end{cases}$$

Theorem 2 shows how tasks are to be scheduled once the number of queues to schedule is determined. If $SQ$ is satisfied then all tasks are scheduled on the shortest queue whereas if $BQ$ is satisfied then tasks are assigned to each queue in a way that tends to equalize their final lengths or in other words to evenly balance the load across both processors. This feature of the allocation (11) can be seen as follows: First $\Delta n$ tasks are given to the shortest queue (queue 1) making it even with the largest queue (i.e. $n_1 + \Delta n = n_2$). The remaining $b - \Delta n$ tasks
are then evenly distributed among the two queues, \( \left\lfloor \frac{b-\Delta n}{2} \right\rfloor \) being given to queue 1 and \( \left\lceil \frac{b-\Delta n}{2} \right\rceil \) to queue 2. As another way to look at this, we observe that the state \( n^* \) resulting from the allocation (11), i.e.,

\[
n^* \equiv (n_1 + l_1(b; n), n_2 + l_2(b; n)),
\]

satisfies

\[
\Delta n^* = \left\lfloor \frac{b-\Delta n}{2} \right\rfloor - \left\lceil \frac{b-\Delta n}{2} \right\rceil = 0, 1.
\]

If SQ is satisfied then it is clear that the minimum expected job response time is obtained when the shortest queue is assigned all \( b \) tasks and thus the expected response time is given by \( ES_{n_1+b}^1 = (n_1 + b)\overline{X} \). Combining Theorem 2 with (8), we thus see that BQ (resp. SQ) is satisfied in state \( n \) when scheduling \( b \) tasks provided

\[
E \max_{k=1,2}\{S_{n_1+k(b; n)}^k\} \leq (\text{resp.} \geq) ES_{n_1+b}^1 = (n_1 + b)\overline{X}.
\]

In view of Claim 1 of Theorem 2 it seems more appropriate to read SQ as meaning shortest queue rather than single queue as previously defined and we adopt this nomenclature in the rest of the paper.

**Proof of Theorem 2.** Claim 1 follows trivially from the assumption \( n_1 \leq n_2 \) since all tasks are i.i.d. r.v.'s. To prove Claim 2, we consider a state \( n \) in \( N^2 \) such that \( 0 \leq n_1 < n_2 \), and assume BQ to be satisfied in state \( n \), in which case necessarily \( 2 \leq b \) and \( \Delta n < b \) by Lemma 1 (and the remark following its proof). The two cases \( b = 2 \) and \( b > 2 \) are considered separately:

If \( b = 2 \), then \( B(b) \) consists of the single allocation \((1, 1)\), which is of course optimal when BQ is satisfied. Here the condition \( \Delta n < b \) implies that the state \( n \) must satisfy either \( \Delta n = 0 \) or \( \Delta n = 1 \), and a direct inspection of (11) shows that \( l_k(b; n) = 1, k = 1, 2 \) in all cases thus establishing the result.

Now assume \( b > 2 \), and observe that \( B(b) \) can be reparametrized as

\[
B(b) = \{(j, b-j) : j = 1, 2, \ldots, b-1\}.
\]

With this in mind, we consider an integer \( j, j = 1, 2, \ldots, b-1 \), such that \( n_1 + j < n_2 + b - j \). Upon applying Proposition A.1 to the state \((n_1+j, n_2+b-j)\), we conclude that

\[
E \max\{S_{n_1+j+1}^1, S_{n_2+b-(j+1)}^2\} \leq E \max\{S_{n_1+j}^1, S_{n_2+b-j}^2\}
\]

(16)
where the vectors \((j, b - j)\) and \((j + 1, b - (j + 1))\) are each in \(B(b)\) provided \(b - j > 0\) and \(b - (j + 1) > 0\), respectively. We can view the passage from allocation \((j, b - j)\) to allocation \((j + 1, b - (j + 1))\) as a transfer operation (with the understanding here that one task has been transferred from queue 1 to queue 2). Therefore, (16) reads as saying that a transfer from allocation \((j, b - j)\) to allocation \((j + 1, b - (j + 1))\) decreases the expected job response time provided that certain constraints on \(j\) are satisfied. Our proof of Claim 2 rests on this observation.

We note that starting with \(j = 1\), we can have at most \(j^*\) transfers so as to satisfy the index constraints of (16) where \(j^*\) is defined by

\[
    j^* = \max\{j : 1 \leq j \leq b - 1, b - (j + 1) > 0, n_1 + j < n_2 + b - j\}. \tag{17}
\]

Repetitively applying the transfer operations yields the following chain of inequalities

\[
    E \max\{S_{n_1+2, 2}^1, S_{n_2+b-2}^2\} \geq \ldots \geq E \max\{S_{n_1+j^*, 2}^1, S_{n_2+b-j^*}^2\} \geq E \max\{S_{n_1+j^*+1, 2}^1, S_{n_2+b-(j^*+1)}^2\}. \tag{18}
\]

The last allocation in this series, namely \(b^* \equiv (j^* + 1, b - (j^* + 1))\), results in the state \(m^* \equiv (n_1 + j^* + 1, n_2 + b - (j^* + 1))\). We claim that this allocation \(b^*\) is optimal in state \(n\) when scheduling the \(b\) tasks on both processors in that it yields the smallest expected response time among all allocations in \(B(b)\). In the process of establishing this optimality claim, we shall see that the allocation \(b^*\) may not necessarily coincide with the allocation given in (11). As shown below, however, they are equivalent in that they incur identical expected response times, so that the optimality of (11) follows from the optimality of \(b^*\). We first establish the response time equivalence of the allocations of \(b^*\) and (11) and, then prove that allocation \(b^*\) is optimal.

To establish the response time equivalence of both allocations first observe from (17) that

\[
    j^* = \max\{j : 1 \leq j \leq b - 2, 2j < \Delta n + b\} \tag{19}
\]

so that two basic cases need to be considered in evaluating \(j^*\).

First assume \(\Delta n + b = 2a\) for some integer \(a\), in which case \(j^* = \min\{b - 2, a - 1\}\). From the conditions \(\Delta n < b\) and \(2 < b\), we get \(2 < 2a < 2b\), whence \(2 \leq a \leq b - 1\), and the conclusion \(j^* = a - 1\) follows. The allocation \(b^*\) for this case is thus \((a, b - a)\) which coincides with (11)
since \( l_1(b; n) = a \) and \( l_2(b; n) = b - a \) by direct inspection. Note in passing that for this case \( m^*_2 - m^*_1 = \Delta n + b - 2a = 0 \).

Assume now \( \Delta n + b = 2a + 1 \), in which case \( j^* = \min \{b - 2, a\} \). Since \( \Delta n < b \) and \( 2 < b \), we have \( 1 < 2a < 2b - 1 \), or equivalently \( 1 \leq a \leq b - 1 \):

If \( a = b - 1 \), then \( j^* = b - 2 = a - 1 \) so that \( b^* = (a, b - a) \). Direct inspection again shows that \( l_1(b; n) = a \) and \( l_2(b; n) = b - a \) since \( \Delta n = 2a + 1 - b = a \) and \( b - \Delta n = a + 1 - a = 1 \). This time we have \( m^*_2 - m^*_1 = \Delta n + b - 2a = 1 \).

On the other hand, if \( 1 \leq a \leq b - 2 \), then \( j^* = a \) so that \( b^* = (a + 1, b - (a + 1)) \), which is different from (11) since \( l_1(b; n) = \lceil \frac{b^* + \Delta n}{2} \rceil = a \) and \( l_2(b; n) = b - a \). However, we note that the allocation \( b^* \) results in state \( m^* = (n_1 + a + 1, n_2 + b - (a + 1)) = (n_1 + a + 1, n_1 + a) \), whereas the allocation (11) results instead in state \( n^* = (n_1 + a, n_2 + b - a) = (n_1 + a, n_1 + a + 1) \). Since the maximum operation is symmetric in its arguments, we see that both allocations have the same expected job response time. We also observe that here \( m^*_2 - m^*_1 = \Delta n + b - 2(a + 1) = -1 \).

Reviewing the analysis, we have thus established the response time equivalence of \( b^* \) and (11) and in passing also that \( |m^*_2 - m^*_1| \leq 1 \). We next show that the allocation \( b^* \) is indeed optimal. To do so, we note that any allocation \( (i, b - i) \) in \( B(b) \) with \( j^* + 1 < i \leq b - 1 \) has the same expected response time as the symmetric allocation \( (j, b - j) \) in \( B(b) \), where \( 1 \leq j \leq j^* + 1 \) and \( n_1 + i = n_2 + b - j \), i.e., \( j = \Delta n + b - i \). The optimality of the allocation \( b^* \) is then an immediate consequence of (18) and of this observation since \( |m^*_2 - m^*_1| \leq 1 \).

4 Parallel versus Sequential Processing

Theorem 2 points to two policies which are naturally associated with the conditions BQ and SQ, respectively. In terms of the underlying parallel processing model, these two policies correspond to sequential and parallel processing, respectively, and the tradeoff between these policies depends heavily on properties of \( X \) and of the maximum operator. One might think that if the number of tasks were larger than the difference between the queue lengths, i.e., \( b > \Delta n \), then it would be beneficial to distribute some of the tasks to the larger queue thus gaining the potential benefits of parallel processing. This policy is optimal if \( \sigma_X^2 = 0 \) (i.e., if \( X \) is deterministic). However, as \( \sigma_X^2 \) increases, the maximum \( \max_{k=1,2} \{S_{n_k + t_k(b,n)}^k\} \) also
“increases” (i.e., becomes more variable) and therefore it might not be beneficial to incur potential delays associated with synchronizing with the second queue. Indeed, as shown in Theorem 5 below, for each value of \( b \), there exists an integer \( n^*(b) \) such that if the queue lengths satisfy \( n_k > n^*(b), k = 1, 2 \), then sequential processing is optimal. In practical systems, the r.v. \( B \) is bounded and for system loads resulting in large queues, Theorem 5 thus implies that the optimal scheduler would most frequently route complete jobs to the shortest queue and thus do very little parallel processing. This result runs counter to the intuition that parallel processing or load sharing is beneficial.

We next establish a monotonicity property of \( \Psi \).

**Lemma 3.** (Monotonicity in \( b \)) Consider a state \( n \) with \( n_1 \leq n_2 \). Then there exists an integer value \( b^* \equiv b^*(n) \geq 0 \) such that in state \( n \),

1. \( \text{BQ} \) is satisfied for \( b \geq b^* \);
2. \( \text{SQ} \) is satisfied for \( b < b^* \).

**Proof.** We prove the lemma by showing that if \( \text{BQ} \) is optimal for \( b \), then it is also optimal for \( b + 1 \). If \( \text{BQ} \) is optimal for \( b \), then

\[
E \max_{k=1,2} \{ S_{n_k+i_k(b+1;n)}^1 \} \leq ES_{n_1;b+1} = (n_1 + b)\bar{X}.
\]

The value of the expected job response time for scheduling \( b + 1 \) tasks under \( \text{BQ} \) is given by \( E \max_{k=1,2} \{ S_{n_k+i_k(b+1;n)}^k \} \). We set

\[
\delta_k(b; n) \equiv \begin{cases} 1, & l_k(b; n) \neq l_k(b + 1; n), \\ 0, & l_k(b; n) = l_k(b + 1; n), \end{cases} \quad k = 1, 2,
\]

and observe from (11) that exactly one of the quantities \( \delta_k(b; n), k = 1, 2 \), is non-zero. Moreover, for each \( k = 1, 2 \), \( l_k(b; n) \) and \( l_k(b + 1; n) \) differ by at most one.

With these facts in mind, we can then write

\[
E \max_{k=1,2} \{ S_{n_k+i_k(b+1;n)}^k \} = E \max_{k=1,2} \{ S_{n_k+i_k(b;n)}^k + \delta_k(b; n)X_{l_k(b;n)+1}^k \}
\]
\[
\begin{align*}
&\leq \ E \max_{k=1,2} \left\{ S_{n_k+1}^k + \delta_k(\delta; n)X_i^k(\delta; n)\right\} \\
&\leq \ (n_1 + \delta)\bar{X} + \bar{X} = ES_{n_1+b+1}^k. \\
\end{align*}
\]

The first inequality follows from the subadditivity of the maximum operator, whereas the last equality results from (20).

We proceed with a monotonicity property of the SQ policy.

**Lemma 4. (Monotonicity for SQ)** Consider a state \( n \) with \( n_1 \leq n_2 \). If SQ is satisfied in state \( n \) when scheduling \( b \geq 1 \) tasks, then

1. SQ is satisfied in state \( (n_1, n_2 + 1) \);

2. SQ is satisfied in state \( (n_1 - 1, n_2) \) (with \( n_1 \geq 1 \)).

**Proof.** (Claim 1.) By assumption we know that

\[
ES_{n_1+b}^1 \leq E \max_{k=1,2} \left\{ S_{n_k+1}^k \right\}. \tag{23}
\]

By direct inspection, with \( n' = (n_1, n_2 + 1) \), we find that \( l_k(b; n) \leq l_k(b; n') \), \( k = 1, 2 \), so that

\[
\max_{k=1,2} \left\{ S_{n_k+1}^k \right\} \leq \max_{k=1,2} \left\{ S_{n_k+1}^k \right\} \tag{24}
\]

by the monotonicity of the maximum operator. Taking expectations in this last inequality, we get

\[
E \max_{k=1,2} \left\{ S_{n_k+1}^k \right\} \leq E \max_{k=1,2} \left\{ S_{n_k+1}^k \right\}. \tag{25}
\]

and the result is now an immediate consequence of (23) and (25).

(Claim 2.) We first observe that the optimal policy in state \( n \) with \( b \) tasks is the same as the optimal policy in state \( (n_1 - 1, n_2) \) with \( b + 1 \) tasks in view of the fact that the expected response times under SQ (resp. BQ) are identical in these cases. Claim 2 now follows from Lemma 3 since if SQ is optimal for \( (n_1 - 1, n_2) \) with \( b + 1 \), then it is also optimal for this state with \( b \) tasks.

The second key structural property of the optimal scheduling policy \( \Psi \) is contained in the next proposition.
Theorem 5. For each integer \( b \geq 1 \) there exists a non-negative integer \( n^*(b) \) such that SQ is optimal for scheduling \( b \) tasks when in states \( n \) satisfying \( n_k \geq n^*(b), \; k = 1, 2 \).

We note here that Theorem 5 holds for all service distributions that have finite variance that is non-zero. We establish it with the help of the next two technical lemmas which together imply the result for "balanced" system states. More precisely, consider a given integer \( b \geq 1 \) and observe that for states of the form \( n = (m, m) \), we have \( l_1(b; n) = \lfloor \frac{b}{2} \rfloor \) and \( l_2(b; n) = \lceil \frac{b}{2} \rceil \) for each \( m = 0, 1, \ldots \). With this in mind, we set

\[
l_1(b) = \lfloor \frac{b}{2} \rfloor \quad \text{and} \quad l_2(b) = \lceil \frac{b}{2} \rceil, \; b = 1, 2, \ldots
\]  

and define

\[
g_S(m; b) = E S_{m+b}^k = (m + b)X, \quad k = 1, 2
\]

\[
g_P(m; b) = E \max_{k=1,2} \{ S_{m+l_k(b)}^k \},
\]

for all \( m = 0, 1, \ldots \) and all \( b = 1, 2, \ldots \). These functions represent the expected job response time for sequential processing (\( g_S \)) and parallel processing (\( g_P \)), respectively.

Lemma 6. For each fixed integer \( b \geq 1 \), the mappings \( m \rightarrow g_S(m; b) \) and \( m \rightarrow g_P(m; b) \) are both increasing and satisfy

\[
X = g_S(m + 1; b) - g_S(m; b) \leq g_P(m + 1; b) - g_P(m; b), \; m = 1, 2, \ldots
\]

so that \( m \rightarrow g_P(m; b) - g_S(m; b) \) changes sign at most once.

Proof. The statements on \( m \rightarrow g_S(m; b) \) follow from its definition. To prove the remaining part, we set

\[
\Delta(m; b) = \max_{k=1,2} \{ S_{m+1+l_k(b)}^k \} - \max_{k=1,2} \{ S_{m+l_k(b)}^k \}, \; m = 1, 2, \ldots
\]

and note that

\[
g_P(m + 1) - g_P(m) = E \Delta(m; b), \; m = 1, 2, \ldots
\]

The proof thus follows if we can show that \( E \Delta(m; b) \geq X \). To prove this, we note that on the event \( \{ S_{m+l_1(b)}^1 \leq S_{m+l_2(b)}^2 \} \), we have

\[
\Delta(m; b) = \max_{k=1,2} \{ S_{m+l_k(b)}^k + X_{m+1+l_k(b)}^k \} - S_{m+l_2(b)}^2
\]
\[
\begin{align*}
&= \max\{S_{m+1}^{1} - S_{m+2}^{2} + X_{m+1}^{1} + X_{m+1}^{2} \}, \\
&\geq X_{m+1}^{2}.
\end{align*}
\] (32)

Since the event \([S_{m+1}^{1} \leq S_{m+2}^{2}\)] is independent of the r.v. \(X_{m+1}^{2}\), we conclude from (32) that
\[
E1[S_{m+1}^{1} \leq S_{m+2}^{2}] \Delta(m; b) \geq P[S_{m+1}^{1} \leq S_{m+2}^{2}] \bar{X}.
\] (33)

By similar arguments, we obtain the inequality
\[
E1[S_{m+2}^{2} < S_{m+1}^{1}] \Delta(m; b) \geq P[S_{m+2}^{2} < S_{m+1}^{1}] \bar{X}.
\] (34)

The proof that \(E \Delta(m; b) \geq \bar{X}\) then follows by adding the inequalities (33) and (34) term by term.

**Lemma 7.** For each fixed integer \(b \geq 1\), there exists a non-negative integer \(n^*(b) \geq 1\) such that
\[
\begin{align*}
g_S(m; b) &\geq g_P(m; b) \text{ if } 0 \leq m < n^*(b) \\
g_S(m; b) &< g_P(m; b) \text{ if } n^*(b) \leq m.
\end{align*}
\] (35) (36)

**Proof.** In view of Lemma 6 it suffices to show that \(g_P(m; b) - g_S(m; b) > 0\) for some finite \(m\).

This will be established by showing that
\[
\lim_{m \to \infty} \frac{1}{\sqrt{m}}[g_P(m; b) - g_S(m; b)] > 0.
\] (37)

From the definition of \(g_S(m; b)\), we first note that
\[
\frac{1}{\sqrt{m}}[g_P(m; b) - g_S(m; b)] = E \max_{k=1,2} \left\{ \frac{S_{m+1}^{k} - (m + b) \bar{X}}{\sqrt{m}} \right\}, \quad m = 1, 2, \ldots
\] (38)

We then write
\[
\frac{S_{m+1}^{k} - (m + b) \bar{X}}{\sqrt{m}} = S_{m}^{k} \cdot \frac{m + l_k(b)}{m} - \frac{(b - l_k(b)) \bar{X}}{\sqrt{m}}, \quad k = 1, 2, \quad m = 1, 2, \ldots
\] (39)
with the simplifying notation

\[ S_m^k \equiv \frac{S_{m+l_k(b)} - (m + l_k(b))\overline{X}}{\sqrt{m + l_k(b)}}, \quad k = 1, 2, \quad m = 1, 2, \ldots \] (40)

From the Central Limit Theorem ([8], p. 169), as \( m \uparrow \infty \), we have

\[ (\vec{S}_m^1, \vec{S}_m^2) \Rightarrow (U^1, U^2) \] (41)

where \( \Rightarrow \) denotes convergence in distribution, and the r.v.’s \( \{U^1, U^2\} \) are i.i.d. normal r.v.’s with zero mean and variance \( \sigma_X^2 \). For each \( k = 1, 2 \), we have \( E|\vec{S}_m^k|^2 = \sigma_X^2 \) for all \( m = 1, 2, \ldots \), so that the r.v.’s \( \{\vec{S}_m^k, m = 1, 2, \ldots\} \) are uniformly integrable ([6], p. 32).

Combining (39) and (41), we readily conclude, as \( m \uparrow \infty \), that

\[ \left( \frac{S_{m+l_1(b)}^1 - (m + b)\overline{X}}{\sqrt{m}}, \frac{S_{m+l_2(b)}^2 - (m + b)\overline{X}}{\sqrt{m}} \right) \Rightarrow (U^1, U^2) \] (42)

whence

\[ \max_{k=1,2} \left\{ \frac{S_{m+l_k(b)}^k - (m + b)\overline{X}}{\sqrt{m}} \right\} \Rightarrow \max \{U^k\}. \] (43)

By a remark made earlier, the r.v.’s involved in this last convergence are also uniformly integrable, and invoking Theorem 5.4 of ([6], p. 32) we get

\[ \lim_{m \to \infty} \frac{1}{\sqrt{m}} [g_P(m; b) - g_S(m; b)] = E \max_{k=1,2} \{U^k\}. \] (44)

From Proposition B.1 (with \( K = 2 \)), we have \( E \max_{k=1,2} \{U^k\} > 0 \), and the proof of (37) is now complete.

Proof of Theorem 5. Lemma 7 gives the result for balanced states of the form \((m, m)\), and Part 1 in Lemma 4 yields the final conclusion.

5 Comparison Results and Discussion

In this section we present results from a set of simulation experiments which we performed in order to determine various response time properties of the optimal scheduler. Throughout,
we make the following assumptions: Arrivals to the system come from a Poisson point source with intensity $\lambda$. Task service times are exponentially distributed with parameter $\mu$; this assumption allows us, as mentioned in section 2, to avoid difficulties arising from residual service times. Moreover, the number of tasks composing each job is determined randomly, and this independently of other random events occurring in the system, past, present and future. Let $B$ be the r.v. denoting the generic number of tasks composing a job, with $\beta_b \equiv P[B = b]$, $b = 1, 2, \ldots$; the expected number of tasks is then given by $\overline{B} = \sum_{b=1}^{\infty} b \beta_b$. Clearly, for this model, the utilization of the system is given by

$$\rho \equiv \frac{\lambda \overline{B}}{2\mu}. \quad (45)$$

When $B = b$, for some fixed integer $b$, then we get $\rho = \lambda b / 2\mu$. All simulation and computational results displayed below are carried out in the generic case where $\mu = 1$, i.e., task service times are exponentially distributed with unit mean.

Our main interest in these simulation experiments consists in determining the performance improvements that the optimal policy yields over simpler policies and to determine the fraction of time the optimal policy uses multiple processors. Here, we implement the optimal policy dynamically as described in section 2. The simple policies that we consider are the AQ policy, which corresponds to the usual practice of distributing the load evenly among all the processors, the SQ policy which assigns the entire job to the processor with the shortest queue size and the round robin policy RR which alternates between queues, i.e. it schedules the tasks composing job $n$ on processor 1 for $n$ even and on processor 2 for $n$ odd. When considering the AQ policy, we always assume that $B = 2J$ for some integer-valued r.v. $J \geq 1$, and $J$ tasks are scheduled on each one of the two queues. The information required by each of the policies differs in that the optimal policy requires knowledge of all the queue lengths whereas the AQ policy uses no knowledge at all, the SQ policy requires only knowing the queue(s) of minimal queue length and the RR policy only requires knowing the last queue scheduled. It is not too surprising that our results show each of the policies have regions over which they perform well. What is somewhat surprising, however, is that the AQ policy can perform very poorly in high utilizations and this argues against the intuition that parallel processing is beneficial. In fact, as we will shortly see, a job is scheduled on only a single processor by the optimal schedule in medium to heavy loads. This conclusion carries over to the case of an arbitrary number of processors [26] where more
extensive results for a general model shows that a job is scheduled on only a small number of processors by the optimal schedule in medium to heavy loads.

In what follows, the performance of these four scheduling policies is expressed through its expected response time in steady-state. A moment of reflection should convince the reader that under all four policies, statistical equilibrium will be realized under the same stability condition, namely \( \rho < 1 \), a condition always enforced thereafter. Determining the expected response time of the optimal schedule policy appears to be mathematically intractable. Even for the simple policies AQ and SQ, the expected response times can only be obtained by simulation or approximation; such approximations have been developed for the AQ policy (with \( B = K \geq 2 \)) in [32, 39], and for the SQ policy in [33, 34]. A special case of the AQ policy for \( B = K = 2 \) does, however, have an exact solution for the expected response time [14, 32] which is given by

\[
T_2 = \frac{12\mu - \lambda}{8} \frac{1}{\mu - \lambda}.
\]  

(46)

The RR policy can be analyzed as a \( E_2/G/1 \) queue which could be analyzed using matrix geometric methods for service times that have a phase distribution [35]. Even though the distributions we consider here are phase distributions, it was more convenient for us in this study to use simulated values for this policy. There is little loss of accuracy in using simulated values, however, since all the simulated values have 99% confidence intervals (using regenerative simulation) that are within 2% of the simulated value. To avoid a cluttered presentation, we do not show points or confidence intervals on the plots of this section, but join simulated values with straight lines. The smoothness of the curves presented in this section attests to the accuracy of the simulation points. Ratio curves are sometimes jagged since small variations in the denominator can result in large variations in the ratio.

To determine the optimal policy, we need a method to calculate the expected value of the maximum of a two independent sums of i.i.d. service times. The quantity we need to evaluate when both queues are scheduled is

\[
f(n) \equiv E \max_{1 \leq i \leq 2} \{ S_{n_i}^i \}, \quad n \in \mathbb{N}^2,
\]  

(47)

where under the assumptions made, \( S_{n_i}^i \) is an Erlang r.v. with mean \( n_i, i = 1, 2 \), since \( \mu = 1 \).
An expression for the quantity (47) is derived in Appendix A.1 and is given by (56) as

\[
f(n) = n_1 + n_2 - \sum_{i=0}^{n_1-1} \sum_{j=0}^{n_2-1} \binom{i+j}{j} \frac{1}{2^{i+j+1}}, \quad n \in \mathbb{N}^2. \tag{48}
\]

<table>
<thead>
<tr>
<th>Experiment</th>
<th>( B )</th>
<th>( \bar{B} )</th>
<th>( \rho \rightarrow 0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>( \beta_2 = 1 )</td>
<td>2</td>
<td>1.5 = f(1,1)</td>
</tr>
<tr>
<td>II</td>
<td>( \beta_4 = 1 )</td>
<td>4</td>
<td>2.75 = f(2,2)</td>
</tr>
<tr>
<td>III</td>
<td>( \beta_2 = 1/2, \beta_6 = 1/2 )</td>
<td>4</td>
<td>2.71 = (1/2)(f(1,1) + f(3,3))</td>
</tr>
<tr>
<td>IV</td>
<td>( \beta_2 = 2/3, \beta_6 = 1/3 )</td>
<td>4</td>
<td>2.69 = (2/3)f(1,1) + (1/3)f(4,4)</td>
</tr>
<tr>
<td>V</td>
<td>( \beta_2 = 1/3, \beta_4 = 1/3, \beta_6 = 1/3 )</td>
<td>4</td>
<td>2.72 = (1/3)(f(1,1) + f(2,2) + f(3,3))</td>
</tr>
</tbody>
</table>

We consider five experiments that differ in the number of tasks that compose a job. Experiment I considers \( B = 2 \) and thus, as mentioned above, the exact value for the expected response time of the AQ policy is available in that case, and is given by (46). For this case, it is interesting to notice that under the optimal policy, both queues are scheduled only when the queue lengths satisfy \( n_2 = n_2 = n \) for \( n = 0, 1, 2 \). In all other states the optimal policy schedules only the shortest queue. In figure 2 we plot the expected response times as a function of \( \rho \) and in figure 3 we plot ratios of expected response times for both the AQ, SQ and RR policies to that of the optimal policy. It is interesting to observe the regions where each one of these policies does well in comparison to the optimal policy. For low utilizations scheduling both queues is close to optimal whereas scheduling only one queue is close to optimal for high utilizations. The best way to explain these relationships is to plot the expected number of queues scheduled by the optimal policy (as done in figure 4). This shows how the optimal policy changes from a regime of using parallel processing for low loads to a regime of using sequential processing for high loads. Since the policies AQ and SQ mimic the operation of the optimal policy for low and high loads, respectively, it is not surprising that their expected response times become close for these regions. Based on ratios of expected response time only, it would be tempting to select the SQ policy over the AQ policy if the optimal policy was not utilized. This is especially true if one considers that for low utilizations, the ratios are for low expected response times whereas for high utilizations, where the policy SQ does very well, these ratios are for large values. Typically in most systems, it is more important to have
good performance for high utilizations since users of the system experience changes in absolute not relative response times and these absolute changes are low for small response times. If information cannot be obtained to determine the shortest queue, then figure 3 shows that the RR policy, which requires no such information, is an overall better choice than the AQ policy.

Figure 5 compares the optimal policy to that of a single server system with a capacity equal to the number of servers. Specifically, since the distributed system has 2 servers, we consider a single server system which delivers 2 units of service per unit time, where jobs are executed completely on this single server. We assume that $B = b$ and thus that the service time duration is the sum of $b$ i.i.d. exponential r.v.'s, and is thus an Erlang r.v. with $b$ stages. With the help of the Pollaczek–Khinchin formula [18], the resulting response time can be written as

$$T_{M/E_b/1} = \frac{1}{2} \left( b - \frac{(b-1)}{2} \rho \right) \frac{1}{\mu(1-\rho)}. \tag{49}$$

The figure plots the ratio of this value to the expected response time for the optimal policy. Clearly the single coalesced server has better response time.

We recall from Theorem 5 that for, $B = b$ and queue lengths greater than $n^*(b)$, the optimal policy schedules all $b$ tasks on the shortest queue. Using (48), we see that the value of $n^*(b)$ can be given as

$$n^*(b) = \min \left\{ m \mid m \geq \sum_{i=0}^{m+[b/2]-1} \sum_{j=0}^{m+[b/2]-1} \binom{i+j}{i} \frac{1}{2i+j+1} \right\}. \tag{50}$$

In figure 6 we plot values of $n^*(b)$ as a function of $b$, where points in figure 6 are the calculated values of $n^*(b)$ according to (50) and the curves are the quadratic polynomials that best fit the points in a least squares sense. This polynomial is given by $n^*(b) \approx -5 - .414b + .789b^2$.

We now consider a set of experiments to determine how variations in the number of tasks composing a job influence expected response times. Experiment II is our base case where $B$ is non-random, i.e., $B = 4$. Experiments III, IV and V have an expected number of tasks equal to 4 but differ in the distribution of job size (see Table 1 for a listing of their parameters). In figures 7-10, we plot the ratio of response times with respect to the optimal policy, obtained from the SQ, AQ and RR policies respectively for experiments II through V. All sets of curves are qualitatively similar, and again show that the performance of the AQ policy is similar to the
of the optimal policy for low utilizations, and that of the SQ policy is similar to the optimal policy for high utilizations. It is clear that the expected response time of the RR policy is always greater than that of SQ and the figures show that it performs better than AQ for high loads. It is interesting to note, that for high utilizations the response time for AQ can be 50% greater than that of the optimal policy, which again suggests that parallel processing can be harmful under conditions of heavy load.

Although the ratios of these simpler policies to that of the optimal policy are similar for experiments II through V, the absolute response times are markedly different as is shown in figure 11. In this figure we plot the ratio of the response times obtained in experiments III through V to that of experiment II. Here we see that for large utilizations the variation in job size causes response times to increase by more than 15%. Interestingly, the ratio of response times is lower than 1 for small utilizations. To explain this, we note that for $\rho \to 0$, the queues upon job arrival can be assumed empty and the optimal policy thus schedules tasks evenly on both queues. This implies that the response times for the experiments are given by the equations given in Table 1. That variation in job size is beneficial for these cases, is a direct consequence of Jensen's inequality and of the integer-concavity of $k \to f(k,k)$, i.e.,

$$f(k + 1, k + 1) - 2f(k, k) + f(k - 1, k - 1) > 0, \quad k = 1, 2, \ldots$$

(51)

This last property readily follows from (48) by algebraic manipulations.

### 6 Conclusions and Future Research

In this paper we derived the individually optimal scheduling policy for executing jobs consisting of a set of independent tasks on two homogeneous processors. We provided simulation results to compare different scheduling policies when task service times were exponentially distributed. Under these assumptions, our results imply that (i) parallel processing is beneficial for low system utilizations, and that (ii) for moderate to high utilizations, processing all the tasks of a job on a single processor produces smaller expected response times. In particular, the policy that executes all the tasks of a job sequentially on the processor with minimal queue length has a better overall response time than the policy which spreads tasks of a job evenly over all the processors in the system. If information about the shortest queue is difficult to
obtain, then using the round robin policy which does not require information about the queue lengths is a still a good choice for systems in high load. Including overheads due to executing jobs in parallel in our model would only increase the benefits of sequential processing. Our model, which is biased towards parallel processing, therefore strongly indicates that for the architectural assumptions of distributed queues one should only run jobs in parallel in lightly loaded systems. Systems are usually run in conditions of heavy load which strongly suggests that parallel processing can actually be harmful to the system if it is naively used.

The conclusions qualitatively do not depend upon the distribution of task service times nor upon the number of processors. However, for large number of processors and service times which are less variable than the exponential distribution, the benefits of parallel processing will be obtained for higher system utilizations than those found in our results. It will still be the case, however, that as the utilization of the system increases the benefits of parallel processing decrease and eventually a system load will be achieved for which sequential processing is optimal. To summarize our results, for a fixed utilization, parallel processing becomes more beneficial as the distribution becomes less variable and for a given task distribution parallel processing becomes more harmful as the utilization increases. These conclusions are consequences of the theorems proved in this paper and specifically Theorem 5 establishes, for all non-zero variance distributions, that for sufficiently high queue lengths, sequential processing is optimal.

Deriving the socially optimal policy is an interesting, open research problem. Although we do not expect the response time for this policy to be markedly better than that of the individually optimal policy derived here, it would be interesting to quantify this improvements. Finally, determining a way to account for residual service times in the derivation of the individually optimal policy is also of interest. Here difficult mathematical problems arise from the fact that residual times on the different processors may not be independent r.v.'s due to the nature of the scheduling policy.

Acknowledgements. We gratefully acknowledge Laurence Durivault's help in generating the simulation results found in the paper. We also thank C.S. Chang for many useful discussions.

This work was initiated while the first author was a summer visitor at the IBM Thomas J. Watson Research Center, Yorktown Heights. The gracious hospitality of the Systems Analysis Department of S. S. Lavenberg is gratefully acknowledged. Additional Support was provided
by the Systems Research Center and made possible through the National Science Foundation’s Engineering Research Centers Program NSFD CDR 88–03012.

7 Appendices

The results in the appendices that follow contain technical results that are not essential for understanding the main conclusions of the paper.

7.1 Appendix A

Proposition A.1. For any state $n$ in $\mathbb{N}^2$ with $0 \leq n_1 < n_2$, we have

$$ E \max \{S_{n_1+1}^1, S_{n_2-1}^2\} \leq E \max \{S_{n_1}^1, S_{n_2}^2\}. \quad (52) $$

Although Proposition A.1 and its consequences given in section 3 do hold for general task distributions, we have elected to provide a proof (52) only in the case where task distributions are exponential, say with unit parameter, in which case

$$ F_n(t) \equiv P[S_n^k \leq t] = 1 - \sum_{i=0}^{n-1} \frac{t^i}{i!} e^{-t}, \quad t \geq 0, \; k = 1, 2, n = 1, 2, \ldots \quad (53) $$

As previously mentioned, in this exponential case our results are exact since the distribution of residual times seen by an arriving job are identical to the distribution of service times. The general case can be established using techniques of stochastic majorization recently developed by Chang [7]. Presenting a proof along these lines would require substantial preliminary work which lies outside the scope of this paper. Our proof of the exponential case uses standard techniques and keeps the paper self-contained. We also note that the case where task service times have an Erlang distribution readily follows from the validity of (52) in the exponential case.

Proof. Setting

$$ T(n) \equiv E \max \{S_{n_1}^1, S_{n_2}^2\}, \quad n \in \mathbb{N}^2 \quad (54) $$

we see that (52) will follow if we show

$$ T(n_1 + 1, n_2 - 1) \leq T(n_1, n_2), \quad 0 \leq n_1 < n_2. \quad (55) $$
We fix \( n \) in \( \mathbb{N}^2 \). With the help of (53), we easily see that

\[
T(n) = \int_0^\infty (1 - F_n_1(t)F_n_2(t))dt
\]

\[
= \int_0^\infty \left\{ \sum_{i=0}^{n_1-1} \frac{t^i}{i!}e^{-t} + \sum_{j=0}^{n_2-1} \frac{t^j}{j!}e^{-t} - \sum_{i=0}^{n_1-1} \sum_{j=0}^{n_2-1} \frac{t^{i+j}}{i!j!}e^{-2t} \right\} dt
\]

\[
= n_1 + n_2 - \sum_{i=0}^{n_1-1} \sum_{j=0}^{n_2-1} \binom{i+j}{i} 2^{-(i+j+1)} \tag{56}
\]

upon using the well-known identities

\[
\int_0^\infty \frac{t^i}{i!}e^{-t}dt = 1, \quad i = 0, 1, \ldots. \tag{57}
\]

We note that the expression just obtained for \( T(n) \) also holds when components of \( n \) do vanish provided we adhere to the convention that summing over an empty set of terms incurs the value zero. This convention is implicitly enforced throughout in order to avoid discussing separately various boundary cases.

To proceed, we consider only the vectors \( n \) in \( \mathbb{N}^2 \) for which \( 0 \leq n_1 < n_2 \). Writing the components of \( n \) as \( n_1 = m \) and \( n_2 = m + p \) with \( m = 0, 1, \ldots \) and \( p = 1, 2, \ldots \), we define

\[
\Delta(m, p) \equiv T(m, m + p) - T(m + 1, m + p - 1), \quad m = 0, 1, \ldots, \quad p = 1, 2, \ldots, \tag{58}
\]

so that the desired inequalities (55) are now equivalent to

\[
\Delta(m, p) \geq 0, \quad m = 0, 1, \ldots, \quad p = 1, 2, \ldots. \tag{59}
\]

Using the expression (56) derived earlier for \( T(n) \), we are led after some simplifications to the expression

\[
\Delta(m, p) = \sum_{i=0}^{m+p-2} \binom{m+i}{i} 2^{-(m+i+1)}
\]

\[
- \sum_{i=0}^{m-1} \binom{m + p + i - 1}{i} 2^{-(m+p+i)} \tag{60}
\]

for all \( m = 0, 1, \ldots \) and \( p = 1, 2, \ldots \).
Fix \( m = 0, 1, \ldots \). By direct inspection of (60) (with \( p = 1 \)) we see that \( \Delta(m, 1) = 0 \) (as expected from the probabilistic interpretation of the involved quantities). Therefore, (59) will be established if we can show that \( p \to \Delta(m, p) \) is non-decreasing, that is
\[
\Delta(m, p + 1) - \Delta(m, p) \geq 0, \ p = 1, 2, \ldots \tag{61}
\]

From (60) we observe by straightforward calculations that
\[
2^{m+p} (\Delta(m, p + 1) - \Delta(m, p))
\]
\[
= \frac{1}{2^m} \binom{2m + p - 1}{m}
+ \sum_{i=0}^{m-1} \frac{1}{2^{i+1}} \binom{m + p + i}{i} \left( \frac{2(m + p)}{m + p + i} - 1 \right), \ p = 1, 2, \ldots \tag{62}
\]
The conclusion (61) now follows upon noting that the first term in the right handside of (62) is obviously positive, and so is the second term since
\[
\frac{2(m + p)}{m + p + i} - 1 = \frac{m + p - i}{m + p + i} > 0, \ i = 0, \ldots, m - 1, \ p = 1, 2, \ldots \tag{63}
\]

7.2 Appendix B

Proposition B.1. Let \( \{U^k, k = 1, 2, \ldots \} \) be i.i.d. \( \mathcal{R} \)-valued r.v.'s with finite mean. If their common probability distribution function \( F \) is symmetric in the sense that
\[
F(-x) = 1 - F(x), \ x \in \mathcal{R} \tag{64}
\]
then
\[
E \max_{k=1,2,\ldots,K} \{U^k\} \geq 0, \ K = 2, 3, \ldots \tag{65}
\]
Moreover, for each \( K = 2, 3, \ldots \), \( E \max_{k=1,2,\ldots,K} \{U^k\} = 0 \) if and only if \( F = \delta_0 \), where \( \delta_0 \) denotes the point mass distribution at 0.

Note that the symmetry condition (64) holds when the r.v.'s \( \{U^k, k = 1, 2, \ldots \} \) are i.i.d. zero-mean normal r.v.'s as in the proof of Lemma 7.
Proof. Fix $K = 2, 3, \ldots$. Under the foregoing assumptions, it is clear that

$$P[\max_{k=1,2,\ldots,K} \{U^k\} \leq z] = F(z)^K, \quad z \in \mathbb{R}. \quad (66)$$

Consequently, we have ([37], p. 5)

$$E \max_{k=1,2,\ldots,K} \{U^k\} = -\int_{-\infty}^{0} P[\max_{k=1,2,\ldots,K} \{U^k\} \leq z]dz + \int_{0}^{\infty} P[\max_{k=1,2,\ldots,K} \{U^k\} > z]dz$$

$$= -\int_{-\infty}^{0} F(x)^K dx + \int_{0}^{\infty} (1 - F(x)^K) dx \quad (67)$$

and using (64), we find after a change of variables that

$$E \max_{k=1,2,\ldots,K} \{U^k\} = \int_{0}^{\infty} (1 - F(x)^K - (1 - F(x))^n) dx. \quad (68)$$

By calculus we see that $1 - a^K - (1 - a)^K > 0$ for $0 < a < 1$, with equality if either $a = 1$ or $a = 0$. Consequently, $E \max_{k=1,2,\ldots,K} \{U^k\} \geq 0$, and we have $E \max_{k=1,2,\ldots,K} \{U^k\} = 0$ if and only if $1 - F(x)^K - (1 - F(x))^K = 0$ for all $x \geq 0$ (except possibly on a countable subset of the positive real line), a condition which under the symmetry condition (64) is easily seen to be equivalent to $F = \delta_0$. \hfill \blacksquare

References


26


Figure 1. The Model
Figure 2. Experiment I. Expected Response Times

- OPTIMAL POLICY
- ALL QUEUES POLICY
- SHORTEST QUEUE POLICY
- ROUND ROBIN POLICY
Figure 3. Experiment 1. Ratio of Expected Response Times
Figure 4. Experiment I. Expected Number of Processors Scheduled
Figure 5. Experiment I. Ratio of Coalesced Server to Optimal
Figure 6. $n^*$ as a function of $b$
Figure 7. Experiment II. Ratios of Expected Response time
Figure 8. Experiment III. Ratios of Expected Response Time

- **ALL QUEUES TO OPTIMAL POLICY**
- **SHORTEST QUEUE TO OPTIMAL POLICY**
- **ROUND ROBIN TO OPTIMAL POLICY**
Figure 9. Experiment IV. Ratios of Expected Response time
Figure 10. Experiment V. Ratios of Expected Response time

- ALL QUEUES TO OPTIMAL POLICY
- SHORTEST QUEUE TO OPTIMAL POLICY
- ROUND ROBIN TO OPTIMAL POLICY
Figure 11. Ratios of Expected Response time

- EXPERIMENT 3
- EXPERIMENT 4
- EXPERIMENT 5
Research Report

Distributed Parallelism Considered Harmful

Armand M. Makowski
Electrical Engineering Department  
and Systems Research Center  
University of Maryland  
College Park, MD  20742

Randolph Nelson
IBM Research Division  
T. J. Watson Research Center  
Yorktown Heights, NY  10598

LIMITED DISTRIBUTION NOTICE

This report has been submitted for publication outside of IBM and will probably be copyrighted if accepted for 
publication. It has been issued as a Research Report for early dissemination of its contents and will be distributed 
outside of IBM up to one year after the date indicated at the top of this page. In view of the transfer of copyright to 
the outside publisher, its distribution outside of IBM prior to publication should be limited to peer communications 
and specific requests. After outside publication, requests should be filled only by reprints or legally obtained copies 
of the article (e.g., payment of royalties).