Heterogeneous multiprocessor systems with breakdowns: performance and optimal repair strategies

Ram Chakka and Isi Mitrani

Computing Science Department, University of Newcastle upon Tyne, Newcastle upon Tyne, NE1 7RU, UK

Abstract


A model of a system with $N$ parallel processors subject to occasional interruptions of service, and a common unbounded queue fed by a Poisson arrival stream, is analyzed in the steady state. The service, breakdown and repair characteristics may vary from processor to processor. A solution method called spectral expansion is used to determine the joint distribution of the state of the processors and the number of jobs in the queue. The problem of optimizing the repair policy is addressed. The optimal policy is determined in the case when all breakdown rates are equal, and some heuristics for the general case are investigated.

1. Introduction

We are interested in the behaviour of a system where jobs are served by a collection of nonidentical parallel processors, each of which breaks down occasionally and takes some time to be repaired (or replaced). The general topic of modelling systems which are subject to interruptions of service has of course received considerable attention in the literature. Some of the work has concentrated on single-processor models [1, 3, 14, 15, 16], and some on multiprocessor ones, where all the processors are statistically identical [2, 8, 9, 12]. However, very little progress has been made in modelling heterogeneous multiprocessor systems with breakdowns. Although
Markov-modulated queueing systems [13, 17] provide a rather general framework in which this problem may be placed, there are no existing solutions. Some of the difficulty of the analysis stems from the fact that, at any moment in time, the total rate at which service is given depends both on the state of the processors and, to a limited extent, on the number of jobs present.

The contribution of this paper is twofold. First, we employ a rather novel solution method that applies to a class of Markov models of heterogeneous multiprocessor systems. This method, called spectral expansion [10], yields the joint distribution of the set of operative processors and the number of jobs in the system in terms of the eigenvalues and left eigenvectors of a certain matrix polynomial. The approach is readily implementable and is efficient in computing various performance measures. It can thus be recommended as an attractive alternative to the matrix-geometric solution [11]. The model and the spectral expansion solution method are described in Sections 2 and 3, respectively.

The second contribution concerns optimization. If the number of repairmen is less than N, thus restricting the number of processors that can be repaired in parallel, then the repair scheduling strategy (i.e. the order in which broken processors are selected for repair) has an important effect on performance. We study this effect at some length, concentrating on the case of a single repairman (Section 4). An interesting problem which, to our knowledge, has not been addressed before, is to find the repair strategy that maximizes the average processing capacity of the system. A related problem – that of optimizing the utilization of the repairman – was considered by Kameda [6], and was solved in the case when the breakdown rates of all processors are equal. We establish a correspondence between the two problems and hence find the optimal capacity strategy for the same special case.

The general optimal capacity problem still lacks an exact solution. We propose several heuristics and examine their performance empirically (Section 5).

2. The model

Jobs arrive into the system in a Poisson stream at rate $\sigma$, and join an unbounded queue. There are N nonidentical parallel processors numbered 1, 2, ..., N. The service times of jobs executed on processor $i$ are distributed exponentially with mean $1/\mu_i$. However, processor $i$ executes jobs only during its operative periods, which are distributed exponentially with mean $1/\xi_i$. At the end of an operative period, processor $i$ breaks down and requires an exponentially distributed repair time with mean $1/\eta_i$. The number of repairs that may proceed in parallel could be restricted: if so, this is expressed by saying that there are $R$ repairmen ($R \leq N$), each of whom can work on at most one repair at a time. Thus, an inoperative period of a processor may include waiting for a repairman. No operative processor can be idle if there are jobs awaiting service, and no repairman can be idle if there are broken down processors. All
interarrival, service, operative and repair random variables are independent of each other.

If there are more operative processors than jobs in the system, then the busy processors are selected according to some service priority ordering. This ordering can be arbitrary, but it is clearly best to execute the jobs on the processors whose service rates are the highest; this is therefore assumed to be the case (it is further assumed that jobs can migrate instantaneously from processor to processor in the middle of a service). Services that are interrupted by breakdowns are eventually resumed (perhaps on a different processor and hence at a different rate) from the point of interruption. Similarly, if $R < N$ and the repair strategy allows pre-emptions of repairs, then these are eventually resumed from the point of interruption and there are no switching delays.

The model is illustrated in Fig. 1.

The system state at time $t$ can be described by a pair of integers $K(t)$ and $J(t)$ specifying the processor configuration and the number of jobs present, respectively. The precise meaning of "processor configuration", and hence the range of values of $K(t)$, depends on the assumptions. For example, if the processors are identical, then it is only necessary to specify how many of them are operative: $K(t) = 0, 1, \ldots, N$. In the heterogeneous case, if $R = N$, or if the repair strategy is pre-emptive priority or processor sharing, the processor configuration should specify, for each processor, whether it is operative or broken down. This can be done by using a range of values $K(t) = 0, 1, \ldots, 2^N - 1$ (e.g. each bit in the $N$-bit binary expansion of $K(t)$ can indicate
the state of one processor). If, on the other hand, \( R < N \) and the repair strategy is nonpre-emptive, then it is necessary to specify both the processors that are broken down and those among them that are being repaired. That would require an even wider range of values for \( K(t) \).

In general, suppose that there are \( M \) processor configurations represented by the values \( K(t) = 0, 1, \ldots, M - 1 \). The model assumptions ensure that \( X = \{K(t); t \geq 0\} \) is an irreducible Markov process. Let \( A \) be the matrix of instantaneous transition rates from state \( k \) to state \( l \) for this process \( (k, l = 0, 1, \ldots, M - 1; k \neq l) \), with zeros on the main diagonal. The elements of \( A \) depend on the parameters \( \xi_i \) and \( \eta_i \) \((i = 1, 2, \ldots, N)\) and, if \( R < N \), on the repair strategy. For example, consider a system with 3 processors and 1 repairman under a pre-emptive priority repair strategy with priority ordering \((1, 2, 3)\) (i.e. processor 1 has top pre-emptive priority for repair, while processor 3 has bottom priority). There are now 8 processor configurations, \( K(t) = 0, 1, \ldots, 7 \), representing the operative states \((0, 0, 0), (0, 0, 1), \ldots, (1, 1, 1)\), respectively (bit \( i \) is 0 when processor \( i \) is broken, \( i \) when operative). In this case, the matrix \( A \) is given by

\[
A = \begin{bmatrix}
0 & 0 & 0 & 0 & \xi_1 & 0 & 0 & 0 \\
0 & \xi_3 & 0 & 0 & 0 & \xi_1 & 0 & 0 \\
0 & \xi_2 & 0 & 0 & 0 & 0 & \eta_1 & 0 \\
0 & \xi_2 & \xi_3 & 0 & 0 & 0 & 0 & \eta_1 \\
\xi_1 & 0 & 0 & 0 & 0 & \eta_2 & 0 & 0 \\
0 & \xi_1 & 0 & \xi_3 & 0 & 0 & \eta_2 & 0 \\
0 & 0 & \xi_1 & 0 & \xi_2 & 0 & 0 & \eta_3 \\
0 & 0 & 0 & \xi_1 & 0 & \xi_2 & \xi_3 & 0
\end{bmatrix}.
\]

Let also \( D^4 \) be the diagonal matrix whose \( i \)th diagonal element is the \( i \)th row sum of \( A \). The matrix \( A - D^4 \) is the generator matrix of the process \( X \).

Denote by \( p \) the steady-state probability that the processor configuration is \( k \). The row vector \( p = (p_0, p_1, \ldots, p_{M-1}) \) is determined by solving the linear equations

\[
p(A - D^4) = 0; \quad pe = 1,
\]

where \( e \) is the column vector with \( M \) elements, all of which are equal to 1. This solution is not too expensive, even for large values of \( M \), because the matrix \( A \) is very sparse. Having obtained \( p \), one can find the steady-state probability \( q_i \) that processor \( i \) is operative. It is given by

\[
q_i = \sum_{k \in x_i} p_k; \quad i = 0, 1, \ldots, N,
\]

where \( x_i \) is the set of processor configurations in which processor \( i \) is operative. The linear combination

\[
\gamma = \sum_{i=1}^{N} q_i \mu_i
\]
will be referred to as the processing capacity of the system. This is the overall average rate at which jobs can be executed. The system is stable when \( \sigma < \gamma \).

In order to compute performance measures involving jobs, it is necessary to study the two-dimensional Markov process \( Y = \{ [K(t), J(t)]; t \geq 0\} \) with state space \( \{0, 1, \ldots, M-1\} \times \{0, 1, \ldots\} \). Assuming that the stability condition holds, our next task is to compute the steady-state probabilities \( p_{k,j} \) that the processor configuration is \( k \) and the number of jobs in the system is \( j \).

3. Spectral expansion solution

The process \( Y \) evolves according to the following instantaneous transition rates:

(a) from state \( (k,j) \) to state \( (l,j) \) \((k, l=0, 1, \ldots, M-1; j=0, 1, \ldots; l \neq k)\), with rate \( A(k, l) \) (for an example of the matrix \( A \), see (1)),

(b) from state \( (k,j) \) to state \( (k,j+1) \) \((k=0, 1, \ldots, M-1; j=0, 1, \ldots)\), with rate \( \sigma \),

(c) from state \( (k,j) \) to state \( (k,j-1) \) \((k=0, 1, \ldots, M-1; j=1, 2, \ldots)\), with rate \( \beta_{k,j} \), equal to the sum of the service rates of the processors which are operative and busy when the processor configuration is \( k \) and the number of jobs is \( j \).

It is important to note that the transition rates (a) and (b) do not depend on \( j \). The rates (c) may depend on \( j \) when \( j < N \), but cease to do so for \( j \geq N \). In the three-processor example mentioned in the previous section, supposing that \( \mu_1 > \mu_2 > \mu_3 \) (which means that the service priority order is \((1, 2, 3)\)), the values of \( \beta_{k,j} \) are

<table>
<thead>
<tr>
<th>configuration</th>
<th>( \beta_{k,1} )</th>
<th>( \beta_{k,2} )</th>
<th>( \beta_{k,j} ) (( j \geq 3 ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>((0, 0, 0))</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>((0, 0, 1))</td>
<td>( \mu_3 )</td>
<td>( \mu_3 )</td>
<td>( \mu_3 )</td>
</tr>
<tr>
<td>((0, 1, 0))</td>
<td>( \mu_2 )</td>
<td>( \mu_2 )</td>
<td>( \mu_2 )</td>
</tr>
<tr>
<td>((0, 1, 1))</td>
<td>( \mu_2 )</td>
<td>( \mu_2 + \mu_3 )</td>
<td>( \mu_2 + \mu_3 )</td>
</tr>
<tr>
<td>((1, 0, 0))</td>
<td>( \mu_1 )</td>
<td>( \mu_1 )</td>
<td>( \mu_1 )</td>
</tr>
<tr>
<td>((1, 0, 1))</td>
<td>( \mu_1 )</td>
<td>( \mu_1 + \mu_3 )</td>
<td>( \mu_1 + \mu_3 )</td>
</tr>
<tr>
<td>((1, 1, 0))</td>
<td>( \mu_1 )</td>
<td>( \mu_1 + \mu_2 )</td>
<td>( \mu_1 + \mu_2 )</td>
</tr>
<tr>
<td>((1, 1, 1))</td>
<td>( \mu_1 )</td>
<td>( \mu_1 + \mu_2 )</td>
<td>( \mu_1 + \mu_2 + \mu_3 )</td>
</tr>
</tbody>
</table>

The probabilities \( \{p_{k,j}\} \) satisfy an infinite set of balance equations. It is convenient to write these in matrix form, by introducing the row vectors,

\[
x_j = (p_{0,j}, p_{1,j}, \ldots, p_{M-1,j}); \quad j = 0, 1, \ldots,
\]

whose elements represent the states with \( j \) jobs in the system. Also, let \( B_j \) be the diagonal matrix

\[
B_j = \text{diag}(\beta_{0,j}, \beta_{1,j}, \ldots, \beta_{M-1,j}); \quad j = 1, 2, \ldots
\]
We have seen that the index $j$ can be omitted when $j \geq N$:

$$B_j = B; \quad j = N, N+1, \ldots$$

The balance equations can now be written as

$$v_j(D\lambda + \sigma I + B_j) = v_{j-1}\sigma I + v_j A + v_{j+1} B_{j+1}; \quad j = 0, 1, \ldots N-1,$$

where $I$ is the $M \times M$ identity matrix and $v_{-1} = 0$ by definition, and

$$v_j(D\lambda + \sigma I + B) = v_{j-1}\sigma I + v_j A + v_{j+1} B; \quad j = N, N+1, \ldots$$

Equation (8) is a homogeneous vector difference equation of order 2, with constant coefficients. It can be rewritten in the form

$$v_jQ_0 + v_{j+1}Q_1 + v_{j+2}Q_2 = 0; \quad j = N-1, N, \ldots,$$

where $Q_0 = \sigma I$, $Q_1 = A - D\lambda - \sigma I - B$ and $Q_2 = B$. Associated with (10) is the characteristic matrix polynomial $Q(\lambda)$ defined as

$$Q(\lambda) = Q_0 + Q_1\lambda + Q_2\lambda^2.$$

Denote by $\lambda_m$ and $\psi_m$ the eigenvalues and corresponding left eigenvectors of $Q(\lambda)$. In other words,

$$\det[Q(\lambda_m)] = 0; \quad \psi_mQ(\lambda_m) = 0; \quad m = 1, 2, \ldots, d,$$

where $d = \text{degree} \{\det[Q(\lambda)]\}$. We shall focus our attention on the case where all $d$ eigenvalues are single. This is the most interesting case in practice, since the likelihood that a real-life problem of this type will exhibit multiple eigenvalues is negligible (if the coefficients of a random polynomial are sampled from a continuous distribution, then the probability that it will have multiple roots is 0).

The following result was established in [10]:

**Proposition 3.1.** Suppose that $c$ of the eigenvalues of $Q(\lambda)$ are strictly inside the unit disk, while the others are on the circumference or outside. Let the numbering be such that $|\lambda_m| < 1$ for $m = 1, 2, \ldots, c$ and $|\lambda_m| > 1$ for $m = c+1, \ldots, d$. Then any solution of equation (10) which can be normalized to a probability distribution is of the form

$$v_j = \sum_{m=1}^{c} x_m\psi_m\lambda_m^j; \quad j = N-1, N, \ldots,$$

where $x_m (m = 1, 2, \ldots, c)$ are arbitrary complex constants.
Note that, if there are nonreal eigenvalues in the unit disk, then they appear in complex-conjugate pairs. The corresponding eigenvectors are also complex-conjugate. The same must be true for the appropriate pairs of constants $x_m$, in order that the right-hand side of (13) be real. To ensure that it is also positive, it seems that the real parts of $\lambda_m, \psi_m$ and $x_m$ should be positive. Indeed, that is invariably observed to be the case.

So far, we have obtained expressions for the vectors $v_{N-1}, v_N, \ldots$, which contain $c$ unknown constants. Now it is time to consider equations (7), for $j=0, 1, \ldots, N-1$. This is a set of $NM$ linear equations with $(N-1)M$ unknown probabilities (the vectors $v_j$ for $j=0, 1, \ldots, N-2$), plus the $c$ constants $x_m$. However, only $NM - 1$ of these equations are linearly independent, since the generator matrix of the Markov process is singular. On the other hand, an additional independent equation is provided by (9).

Clearly, this set of $NM$ equations with $(N-1)M + c$ unknowns will have a unique solution if, and only if, $c=M$. This observation, together with the fact that an irreducible Markov process has a steady-state distribution if, and only if, its balance and normalizing equations have a unique solution, implies the following proposition.

**Proposition 3.2.** The condition $c=M$ (the number of eigenvalues of $Q(\lambda)$ strictly inside the unit disk is equal to the number of processor configurations) is necessary and sufficient for the stability of the Markov process $Y=\{[K(t), J(t)]; t \geq 0\}$.

The intuitively appealing stability condition given in the previous section, $\sigma < \gamma$, should be equivalent to the one provided by Proposition 3.2. This is confirmed by all numerical experiments, but we have not been able to prove it formally.

A few additional remarks concerning Proposition 3.2 are, perhaps, in order. It can be shown that the eigenvalue of $Q(\lambda)$ with the largest modulus in the unit disk is always real. That is, in fact, the Perron–Frobenius eigenvalue of Neuts’ matrix $R$, defined in [11]. When the load on the system increases, i.e. when $\sigma$ approaches $\gamma$, that eigenvalue approaches 1, while the other eigenvalues in the unit disk remain strictly in the interior. Hence, the heavy-traffic behaviour of the system is governed by just one (real) term in the spectral expansion – the term involving the dominant eigenvalue and its left eigenvector.

In summary, our solution procedure consists of the following steps:

1. Compute the eigenvalues $\lambda_m$ and the corresponding left eigenvectors $\psi_m$ of $Q(\lambda)$. If $c \neq M$, stop; a steady-state distribution does not exist.
2. Solve the finite set of linear equations (7) and (9), with $v_{N-1}$ and $v_N$ given by (13), to find the constants $x_m$ and the vectors $v_j$ for $j < N-1$. The entire joint distribution $p_{k,j}$ is now determined.
3. Use the obtained solution for the purpose of computing various moments, marginal probabilities, percentiles and other system performance measures that may be of interest.

An efficient computational method for step 1, based on reducing the quadratic eigenvalue–eigenvector problem to a linear one, is described in [2, 5].
The computational complexity of step 2 is not as great as it appears. The vectors \( v_{N-2}, v_{N-3}, \ldots, v_0 \) can be expressed in terms of \( v_{M-1} \) and \( v_M \), by successive application of (7) for \( j = N - 1, N - 2, \ldots, 1 \). This leaves just equations (7) for \( j = 0 \), plus (9) (a total of \( M \) independent linear equations) for the \( M \) unknowns \( x_m \).

Some numerical results obtained by the above procedure are presented in the following section.

It should be pointed out that the spectral expansion method is applicable in considerably more general situations than the one considered here. For example, the job arrival rate could depend on the processor configuration. The only effect of this generalization would be to replace the matrix \( Q_0 = \sigma I \) with a different diagonal matrix with nonidentical elements. One could also envisage models where the breakdown rate of a processor depends on whether the latter is busy or not; where breakdowns can be triggered by arrivals; or where several processors are liable to break down or be repaired simultaneously. Assumptions of that kind are easily reflected in the form of the matrices \( Q_0, Q_1 \) and \( Q_2 \).

A more substantial generalization would be to allow arrivals and departures to occur in batches of fixed or random sizes. Provided that the batch sizes are bounded, such models also lead to finite vector difference equations. However, the order of those equations is higher than 2. More precisely, if the number of jobs in the system can jump up by at most \( r_1 \) at a time, and down by at most \( r_2 \) at a time, with or without triggered breakdowns or repairs, then the difference equation similar to (10) would be of order \( r_1 + r_2 \). The degree of the matrix polynomial \( Q(\lambda) \) would correspond to that order. The complexity of the solution procedure would obviously be higher, but its form, based on Propositions 3.1 and 3.2, is the same.

4. Repair strategy optimization

When the number of repairmen is less than the number of processors, the order in which processors are selected for repair has an influence on the performance of the system. Different repair strategies can lead to different values of the processing capacity \( \gamma \) and hence to arbitrarily large differences in performance measures like the average queue size or the average response time. This situation is illustrated in Fig. 2, where the average number of jobs \( E(J) \) in a 3-processor system with one repairman, is plotted against the job arrival rate. Four repair strategies are compared, three of which are of the pre-emptive-resume type, allowing repairs to be interrupted at an arbitrary point without any overheads, and one is non-pre-emptive. In all cases, the values of \( E(J) \) are determined by applying the spectral expansion procedure described in the previous section.

The parameters in this example have not been chosen to be realistic; nevertheless, the behaviour that is displayed in the figure is quite typical. The important feature of all such performance curves is that they have a vertical asymptote at the point where
the arrival rate is equal to the processing capacity. Any difference in $\gamma$, no matter how small, caused by a change in the repair strategy, implies different vertical asymptotes. Hence, given a sufficiently heavy traffic, there will be unbounded differences in performance. In particular, on the interval between the two processing capacities, the queue is saturated under one strategy and stable under the other.
Clearly, it can be very important to use a repair strategy that maximizes the processing capacity. We shall consider the problem of finding such strategies, in the case where \( N \) heterogeneous processors are attended to by a single repairman. This is obviously the case with the greatest practical relevance, since increasing the number of repairmen can only reduce the effect of the repair strategy on the processing capacity.

For a given strategy, let \( u_i \) be the fraction of time that processor \( i \) spends being repaired (\( i = 1, 2, \ldots, N \)); alternatively, this is the repairman utilization factor due to processor \( i \). Let also \( T_i \) be the average period between two consecutive instants when processor \( i \) becomes operative (i.e. the average operative–inoperative cycle for processor \( i \)). Then we can write

\[
 u_i = \frac{1/\eta_i}{T_i}; \quad i = 1, 2, \ldots, N. \tag{14}
\]

Similarly, the probability that processor \( i \) is operative can be written as

\[
 q_i = \frac{1/\xi_i}{T_i}; \quad i = 1, 2, \ldots, N. \tag{15}
\]

Hence, we have the relation \( q_i = (\eta_i/\xi_i)u_i \). This allows us to rewrite the expression for the processing capacity (4) in the form

\[
 \gamma = \sum_{i=1}^{N} c_i u_i, \tag{16}
\]

where \( c_i = \mu_i \eta_i / \xi_i \).

Thus, the problem of maximizing the processing capacity can be reduced to that of maximizing a linear combination of repairman utilizations. This latter problem has been studied by Kameda [6, 7] in the context of finite-source queues with different jobs and a single server. He considered the special case where the job arrival rates (which correspond to our breakdown rates) are equal: \( \xi_1 = \xi_2 = \cdots = \xi_N \). Under that assumption, and using our terminology, Kameda’s main result can be summarized as follows:

(a) When the exact repair times are not known in advance, the strategy that maximizes the right-hand side of (16) is one of the \( N! \) static pre-emptive priority strategies.

(b) In the optimal pre-emptive priority strategy, processor \( i \) is given higher priority for repair than processor \( j \) if \( c_i > c_j \).

Thus, when the breakdown rates are equal, the highest processing capacity is achieved by a static repair strategy which gives processor \( i \) pre-emptive priority over processor \( j \) if \( \mu_i \eta_i > \mu_j \eta_j \).

In the general case of unequal breakdown rates, it is not known whether statement (a) is true or not. It may be that, in order to achieve global optimality, one has to allow dynamic scheduling decisions which depend in a complex way on the current processor configuration. Nevertheless, we shall confine our search to the set of \( N! \) static pre-emptive priority policies. Even so, it seems that there is no simple rule for
determining the optimal allocation of priorities. A straightforward application of the
recipe contained in statement (b) leads to the following heuristic:

**Heuristic 1.** Give processor $i$ pre-emptive priority for repair over processor $j$ if
\[
\mu_i \eta_i / \xi_i > \mu_j \eta_j / \xi_j.
\]

Other heuristics are suggested by considering a two-processor system with arbit-
rary breakdown and repair rates. Let the processing capacities under the two pre-
emptive priority strategies $(1, 2)$ and $(2, 1)$ be $\gamma^{(1, 2)}$ and $\gamma^{(2, 1)}$, respectively. By solving
equations (2) explicitly (there are now 4 processor configurations: $(0, 0), (0, 1), (1, 0)$
and $(1, 1)$), and then using either (4) or (16), we find

\[
\gamma^{(1, 2)} = \frac{\eta_1}{\xi_1 + \eta_1} \left[ \frac{\mu_1 + \mu_2 \eta_2 (s - \eta_2)}{\xi_2 s + \eta_1 \eta_2} \right],
\]

where $s = \xi_1 + \xi_2 + \eta_1 + \eta_2$. The expression for $\gamma^{(2, 1)}$ is similar, with the indices 1 and
2 interchanged. A little further manipulation yields

\[
\gamma^{(1, 2)} - \gamma^{(2, 1)} = \frac{\xi_1 \xi_2 s}{(\xi_1 + \eta_1)(\xi_2 + \eta_2)} \left[ \frac{\mu_1 \eta_1}{\xi_1 s + \eta_1 \eta_2} - \frac{\mu_2 \eta_2}{\xi_2 s + \eta_1 \eta_2} \right].
\]

Thus, when $N = 2$, the optimal priority allocation depends on the quantity

\[
d_{1, 2} = \frac{\mu_1 \eta_1}{\xi_1 s + \eta_1 \eta_2} - \frac{\mu_2 \eta_2}{\xi_2 s + \eta_1 \eta_2}.
\]

Processor 1 should have higher repair priority than processor 2 if $d_{1, 2} > 0$, and lower
priority if $d_{1, 2} < 0$.

The above result suggests ways of ranking the processors in a general $N$-processor
system. Define, for any pair of processors $i$ and $j$, the quantity

\[
d_{i, j} = \frac{\mu_i \eta_i}{\xi_i s_{i, j} + \eta_i \eta_j} - \frac{\mu_j \eta_j}{\xi_j s_{i, j} + \eta_i \eta_j},
\]

where $s_{i, j} = \xi_i + \xi_j + \eta_i + \eta_j$. Also define the indicator

\[
\delta_{i, j} = \begin{cases} 1 & \text{if } d_{i, j} > 0, \\ 0 & \text{if } d_{i, j} \leq 0. \end{cases}
\]

Processor $i$ is said to be preferable to processor $j$ if $\delta_{i, j} = 1$.

Two heuristics based on this notion of preference are the following.

**Heuristic 2.** Assign to processor $i$ an integer weight $w_i$ equal to the number of processors
to which it is preferable:

\[
w_i = \sum_{j=1}^{N} \delta_{i, j}; \quad i = 1, 2, \ldots, N.
\]

Give processor $i$ higher priority than processor $j$ if $w_i > w_j$. 
Heuristic 3. Assign to processor \( i \) a real weight \( w_i \) equal to the accumulated differences (20), for all processors to which it is preferable:

\[
w_i = \sum_{j=1}^{N} \delta_{i,j} d_{i,j}, \quad i = 1, 2, \ldots, N.
\]

Give processor \( i \) higher priority than processor \( j \) if \( w_i > w_j \).

Heuristic 2 concentrates on the existence of preference relations, whereas Heuristic 3 attempts to take into account their extent. Of course, both produce the optimal allocation in the case \( N = 2 \).

The relation preferable is not transitive. For example, it may happen that processor \( i \) is preferable to processor \( j \), processor \( j \) is preferable to processor \( k \), and processor \( k \) is preferable to processor \( i \). Heuristic 2 could then assign equal weights to those three processors, necessitating some tie-breaking rule. We have simply used the processor indices to break ties. It should be pointed out that these occurrences are very rare (less than 1% of all cases examined).

One can put forward other heuristics which, while being intuitively weaker, do not seem unreasonable. For example:

Heuristic 4. Give processor \( i \) priority over processor \( j \) if \( \mu_i \eta_i > \mu_j \eta_j \).

Heuristic 5. Give processor \( i \) priority over processor \( j \) if \( \mu_i > \mu_j \).

Heuristic 6. Give processor \( i \) priority over processor \( j \) if \( \xi_i < \xi_j \).

Heuristic 7. Give processor \( i \) priority over processor \( j \) if \( \eta_i > \eta_j \).

In the absence of theoretical results with general validity, the only way of evaluating all these heuristics is by experimentation. Some empirical results are reported in the next section.

5. Evaluation of the heuristics

The first set of three experiments involves a 100-processor system where the breakdown rates are about an order of magnitude smaller than the repair rates, which are in turn a couple of orders of magnitude smaller than the service rates. An experiment consists of generating 100 random models, i.e. random sets of values for the parameters \( \mu_i, \xi_i \) and \( \eta_i \) \((i = 1, 2, \ldots, 100)\). The latter are uniformly distributed within prescribed ranges. For each model, the processing capacity under the seven heuristics of the previous section, and under the first-come-first-served (FCFS) repair strategy, is estimated by simulation (exact solutions are not feasible, due to the large state spaces of the Markov chains). The three experiments differ in the range of the parameters \( \eta_i \).
Heterogeneous multiprocessor systems with breakdowns

The bar chart in Fig. 3 shows, for each heuristic, the number of models in which it produced the highest processing capacity among the eight heuristics (Heuristic 8 is the FCFS strategy). We observe that Heuristic 1 is most frequently best, while Heuristics 5, 6 and 8 are never best. However, the dominance of Heuristic 1 diminishes when the mean and variance of $\eta_i$ increase, and hence the average utilization of the repairman decreases. A somewhat surprising feature of this figure is the poor performance of Heuristic 3, and the relatively good performance of Heuristic 4.

It is interesting to note the improvement in processing capacity achieved by each heuristic, compared to FCFS. This is shown in Fig. 4, where each bar measures the...
quantity \( (\gamma_{\text{heuristic}} - \gamma_{\text{FCFS}})/\gamma_{\text{FCFS}} \), averaged over the 100 models. It is rather remarkable that, with the exception of 5 and 6, all heuristics achieve roughly the same average relative improvement. More predictable is the fact that the heavier the load on the repairman, the greater the magnitude of that improvement (however, even in the third experiment, a 25% average improvement is achieved by most heuristics).

Another strategy that can be used as a standard of comparison instead of FCFS is the random repair strategy, whereby at every scheduling decision instant, every one of the broken processors is equally likely to be selected for repair. This was tried, but no appreciable difference between the random and the FCFS strategies was observed.
The above results do not tell us how close to optimal are the heuristics, since they are compared only among themselves. On the other hand, in a 100-processor system, an exhaustive search of all permutations in order to find the optimal priority allocation is obviously out of the question. Therefore, we have carried out an experiment with a 6-processor system.

Again, 100 random models are generated, sampling the parameters $\mu_i$, $\xi_i$ and $\eta_i$ ($i = 1, 2, ..., 6$) uniformly from prescribed ranges. Those ranges are $\mu_i \in (5000, 95000)$, $\xi_i \in (0, 30)$, $\eta_i \in (5, 145)$. The breakdown and repair rates are now much closer together, so that there is still some competition for the repairman.

For each model, the processing capacity $\gamma$ is computed (exactly, by solving the appropriate Markov chain) under all $6! = 720$ possible pre-emptive priority allocations. The seven heuristics are of course among them, and so is the optimal allocation. The following performance measures of each heuristic are determined in every model:

- $b =$ number of allocations that are better than the heuristic,
- $h = (\gamma_{\text{optimal}} - \gamma_{\text{heuristic}}) / \gamma_{\text{optimal}}$,
- $a = 100(\gamma_{\text{heuristic}} - \gamma_{\text{FCFS}}) / \gamma_{\text{FCFS}}$.

(The processing capacity of the FCFS strategy is again estimated by simulation.)

A rough comparison between the heuristics is displayed in Table 1, where the quantities $b$, $h$ and $a$ are averaged over the 100 models.

The table shows that in this experiment the first three heuristics perform considerably better than the others. Their average relative distances from the optimum $h$ are very small indeed. However, even the worst two Heuristics 5 and 6 yield processing capacities within about 12% of the optimum. The average relative improvement in capacity with respect to the FCFS strategy is considerable: it is on the order of 30% for all heuristics except 5 and 6.

The frequency histograms for $b$ and $h$ are displayed in Figs. 5 and 6, respectively. It can be seen that Heuristic 2 finds the best allocation in about 63% of the models, Heuristic 3 is optimal in about 45% of the models, and Heuristic 1 is optimal in less than 30% of the models. Figure 6 demonstrates very clearly that, even when the best allocation is not found, the processing capacity by all three heuristics is very close to optimal. In almost 90% of the models for Heuristic 2, 75% for Heuristic 3 and 50% for heuristic 1, the relative error as measured by $h$ is less than 0.001.
A similar experiment with a more heavily loaded repairman produced similar results, except that there the order of the first three heuristics was reversed: 1 was slightly better than 3, which was slightly better than 2. Again, the relative errors of all heuristics apart from 5 and 6 were very small.

6. Conclusions

We have presented an efficient solution method, which can be applied to a class of heterogeneous multiprocessor models of moderate size. The bounds on the feasibility
of a solution are dictated by the number of possible processor configurations: when that number is very large, the eigenvalue-eigenvector problem becomes ill-conditioned.

As far as the optimal repair strategies are concerned, it is clear that any one among four or five heuristics would produce acceptable results in practice. In all cases, the priorities allocated may be pre-emptive or nonpre-emptive. If one accepts the proposition, intuitively supported by Kameda’s argument in [7], that the globally optimal policy is one of the $N!$ pre-emptive priority ones, then any nonpre-emptive policy is bound to be sub-optimal. However, our experiments provide a strong indication that the processing capacities of several sub-optimal strategies tend to be very close to that
of the optimal one. The effort of searching for an optimal allocation would be justified only if the traffic rate is such that the job queue is close to saturation.

The outstanding theoretical problems in this context are

(i) to prove or disprove the fact that, when the exact repair times are not known in advance, the globally optimal policy is indeed one of the $N!$ static pre-emptive priority allocations,

(ii) to find an efficient algorithm for determining the optimal (as opposed to a nearly optimal) priority allocation.

It would also be interesting to get an estimate of the rate at which the relative error of the heuristics grows with the number of processors. Further experimentation could shed some light on this, but it will be expensive (already with 7 processors, there are 5040 possible priority allocations).

Acknowledgment

This work was carried out in connection with the Basic Research projects PDCS II (Predictably Dependable Computer Systems) and QMIPS (quantitative methods in parallel systems), funded by the European Community.

References
