An Empirical Study of the Performance of Distributed Replicated Systems

P.D. Ezhilchelvan
I. Mitrani
S.K. Shrivastava
An Empirical Study of the Performance of Distributed Replicated Systems

P.D. Ezhilchelvan, I. Mitrani, S.K. Shrivastava

Abstract

Replicated processing with majority voting - N modular redundant processing - is a well known method of achieving fault tolerance. We consider a distributed system where a job can be broken into a number of sub-jobs which are processed sequentially at various processors of the distributed system. The performance of such a system is then compared with the replicated version of the system where each sub-job will require concurrent replicated processing with majority voting. The objective is to examine the relative impact of voting times and processor failure probabilities on the overall performance. A variety of system configurations are evaluated by means of computer simulations, and also by analytical approximations. The accuracy of the latter is examined.
Bibliographical details

EZHLICHELVAL, Paul Devadoss

-An empirical study of the performance of distributed replicated systems.
[By] P.D. Ezhilchelvan, I. Mitrani and S.K. Shrivastava

Newcastle upon Tyne: University of Newcastle upon Tyne: Computing Laboratory, 1989.

(University of Newcastle upon Tyne, Computing Laboratory, Technical Report Series, no. 278)

Added entries
MISTRANI, Israel
SHRIVASTAVA, Santosh Kumar
UNIVERSITY OF NEWCASTLE UPON TYNE.
Computing Laboratory. Technical Report Series. 278

Abstract

Replicated processing with majority voting - N modular redundant processing - is a well known method of achieving fault tolerance. We consider a distributed system where a job can be broken into a number of sub-jobs which are processed sequentially at various processors of the distributed system. The performance of such a system is then compared with the replicated version of the system where each sub-job will require concurrent replicated processing with majority voting. The objective is to examine the relative impact of voting times and processor failure probabilities on the overall performance. A variety of system configurations are evaluated by means of computer simulations, and also by analytical approximations. The accuracy of the latter is examined.

About the authors

Mr. Ezhilchelvan has been a Research Associate in the Computing Laboratory since September, 1983.
Dr. Mitrani is a reader with the Computing Laboratory. Professor Shrivastava joined the Computing Laboratory in August, 1975, where he is a Professor.

Suggested keywords

DISTRIBUTED SYSTEMS
MAJORITY VOTING
NMR PROCESSING
PERFORMANCE EVALUATION
SIMULATION

Suggested classmarks (primary classmark underlined)
Dewey (18th): 002.64404 001.6425
U.D.C. 681.322.06 519.687
AN EMPIRICAL STUDY OF THE PERFORMANCE OF DISTRIBUTED REPLICATED SYSTEMS

Paul Ezhilchelvan, Isi Mitran, and Santosh Shrivastava
Computing Laboratory,
The University of Newcastle upon Tyne,
NE1 7RU, UK.

ABSTRACT

Replicated processing with majority voting - N modular redundant processing - is a well known method of achieving fault tolerance. We consider a distributed system where a job can be broken into a number of sub-jobs which are processed sequentially at various processors of the distributed system. The performance of such a system is then compared with the replicated version of the system where each sub-job will require concurrent replicated processing with majority voting. The objective is to examine the relative impact of voting times and processor failure probabilities on the overall performance. A variety of system configurations are evaluated by means of computer simulations, and also by analytical approximations. The accuracy of the latter is examined.

Keywords: Distributed systems, NMR processing, Majority voting, Simulation, Performance evaluation.
1. Introduction

Replicated processing with majority voting - N modular redundant (NMR) processing - provides a powerful means of constructing highly reliable computing systems. This form of processing is of particular relevance to real time systems requiring a very high degree of reliability for two reasons: (i) the capability of masking the effects of failures of processing modules by majority voting means that there need not be a sudden degradation in response times due to failures; and (ii) majority voters are capable of tolerating arbitrary behaviour of failed modules (capable of tolerating Byzantine failures [Lampo82] of processing modules). In this paper we present an analytical and simulation study of the performance of distributed replicated processing. In particular we examine the influence of majority voting times and processor failure rates on the response times of jobs that require processing at a number of different nodes. The system features that are of interest to us are discussed in general terms in section 2. Some of the factors which influence performance are mentioned there. The particular models that are used for the purpose of performance evaluation are described in section 3. Two evaluation methods are employed: analytical approximations and computer simulations. The analysis, which is quite simple but nevertheless effective, is presented in section 4. Section 5 reports on a number of experiments where the models are simulated for different values of the parameters and the simulation results are compared to the analytical approximations. The latter are shown to be capable of predicting performance with sufficient degree of accuracy. Conclusions from this work are drawn in section 6.
2. Distributed replicated processing systems

We start by considering a general distributed processing system without replication. We assume that such a simplex system consists of a number of processors connected by a communication system. Each processor is capable of performing one or more system functions (for example, in an avionics system, such functions could be sensor related processing, flight path related processing and so forth). The environment of the system consists of a set of initiators (the entities that demand services from the system at arbitrary times). A service request from an initiator gives rise to a job which requires processing at one or more processors in some order; at any time there could be several such requests being processed by the system. Suppose that a job requires processing at processors $P_1, P_2, \ldots, P_N$, in that order, before it completes. Then the end-to-end delay for that job - its sojourn time in the system - is composed of waiting and service at $P_1$, followed by a transmission delay from $P_1$ to $P_2$, etc., until service is completed at $P_N$. A message passed from $P_i$ to $P_{i+1}$, containing the relevant state information necessary for processing a job at $P_{i+1}$, will be termed a 'task message'.

The replicated version of the above system is assumed to work as follows. It will be assumed that tolerance to a bounded number of processor failures is required. Let the degree of processor replication be $r$. Then each system function is performed by an ensemble of $r$ processors, referred to as an NMR node. The degree of replication is of the form $r=2m-1$, where $m$ is the minimum number of results that would constitute a majority. Alternatively, $m-1$ is the maximum number of processor failures which can be tolerated within a node. When $r=3$, we get the well known TMR system (triple modular redundant system) capable of tolerating a single processor
failure per node. Consider now the processing of a job in a replicated system where the job needs processing at nodes 1 to N in that order. Each of the r processors in the first node receives a separate version of the job and works on it independently of the others. Those versions will be referred to as 'siblings'. When a sibling is completed by a processor, r copies of the resulting task message are sent to node 2. Thus, each processor in node i, (i > 1), receives r messages from node i - 1. These messages are majority voted by the voter process of the processor; if a majority can be formed, then the voted sibling is processed and r copies of the task message are sent to node i + 1 (except in the case of the last node, where a single task message is sent by each processor to a final voter).

The structure of each processor within a node is illustrated in figure 1.

![Diagram of a processor](image)

**Fig. 1:** Voting and job processing at a processor of an NMR node

Both the voter process and the service process of the processor maintain pools of buffers for storing incoming messages. The voter process performs voting as soon as it can form a majority on a given set of messages received from a sending node. The voted messages are stored in the voted message pool. The service process picks up a voted message from the pool and processes the request associated with it; if further processing at a subsequent node is required, then r copies of the task message are sent on, as
stated earlier.

In general, it should be assumed that processors maintain states which affect the execution of jobs, and that the execution of a job by a processor can modify its state. Job processing is assumed to be deterministic, in the following sense: if two non-faulty processes have identical states and then process two copies of a task message, then the final states of the processes will be identical. Assuming that all non-faulty service processes of a node have identical initial states before any job processing begins, we require that all these processes process voted messages in an identical order. This sequencing requirement is necessary in order to ensure that correctly functioning processors of a node produce identical results. Some form of protocol is required to meet this requirement. For example, the processors of a node could execute an agreement protocol for selecting identical messages from the voted message pool [Manci86]. For a review of techniques for meeting the sequencing requirement, see [Shriv87].

From the above discussion, we can identify a number of factors which may have an impact on the sojourn time of a job within a replicated system:

(i) Voting times: Voting consumes processing resources. If the time taken to reach a majority decision is relatively large compared to the actual processing time, then the sojourn time for a task is likely to be substantially larger than the corresponding time for the simplex system.

(ii) Processor failure rates: In a simplex system, a processor failure cannot be masked (the affected job will not complete). Whilst a replicated system can tolerate a bounded number of failures, such failures can affect sojourn times. Consider, for example, the progress of a job through two consecutive nodes, \( i \) and \( i+1 \). Moreover, suppose that the loads on the processors in these nodes are not identical (e.g., in addition to replicated
processing, each processor in node $i$ may have other, non-replicated processing functions to perform). Thus, the delay times of the job's siblings in node $i$ may well be different. As a consequence, the task messages associated with those siblings arrive at a given voter in node $i+1$ at different times (these differences may be exacerbated by variations in message transmission delays). If there are no failures in node $i$, a voter in node $i+1$ can form a majority as soon as $m$ copies of the task message arrive. On the other hand, processor failures in node $i$ can delay a voter in node $i+1$ by obliging it to wait for other, slower node $i$ processors to respond. Thus, even if failures are masked, this is at the expense of increases in sojourn times.

(iii) Extra message traffic: A replicated system can generate more messages than its non-replicated counterpart. The impact of this increase will depend upon the network bandwidth, topology and architecture. For example, if nodes are connected by $r$-redundant busses, a processor need only send a single task message on each of the $r$ busses - thus a given bus will experience the same message traffic as in the simplex system, so the extra message traffic will have little impact on the sojourn time in this particular case.

(iv) Sequencing overheads: As stated earlier, processors of a node must be kept in step to prevent sequence failures. A sequencing protocol can consume both processing and communication resources, thereby contributing to the sojourn time of tasks.

In this paper we investigate the impact of first two factors on two system performance indicators: the average sojourn time of a successfully completed job (also referred to as the system response time), and a reliability measure called operative time expressed as the fraction of the mission time
during which the system is operative (ideally this should be one).

We shall assume that there is enough communication bandwidth available, so that factor three is of little significance. Moreover, task messages will be assumed to arrive at a destination node in the order in which they are sent by the source processor. This, together with the fact that we will assume our distributed processing system to be a pipeline, with a unique route followed by all jobs, implies that there is no need for special sequencing protocols. It is enough to treat the message pools in figure 1 as FIFO queues. Of course, there is still a need to identify the siblings of a job, so that these can be matched by the voters.

Performance evaluation of distributed replicated systems have not been reported in the literature, although single node systems have been evaluated [York83, Pitte86]. In [York83], the authors evaluate the impact of voting on throughput, both analytically and experimentally. In [Pitte86], a TMR database system has been evaluated experimentally to examine the throughput. In this experiment, the throughput of the TMR system was less than the simplex system since extra message overheads were not negligible, as no redundancy in communication was present.

3. Model definition

We study systems consisting of $N$ nodes, $1, 2, \ldots, N$. Those nodes are visited by every job, in the order in which they are numbered. In the case of a simplex, non-replicated system, each node contains a single processor and an unbounded queue where jobs wait in order of arrival. In the replicated case, all nodes have the same degree of replication, $r$ (typically $r=3$). The structure of an NMR node is illustrated in figure 2.

The voting and computational functions of each replicate are separated
Fig. 2: Model of an NMR node

and are carried out by two independent servers. The latter are referred to as the 'voter' and 'service processor', respectively. There is an unbounded queue for each voter and each service processor. A voter queue in node $i$ ($i > 1$), receives job siblings (or, rather, task messages corresponding to job siblings) from each service processor in node $i-1$. As soon as $m$ siblings of a job are present in the queue, where $m = \lfloor r/2 \rfloor$, the voter may attempt to vote on them. If that vote results in an agreement, then the job is passed on to the service queue; otherwise, additional siblings are awaited and the procedure is repeated after every new arrival. If, after all $r$ siblings are present, there is still no agreement, then the job is effectively discarded.

Each service processor services the jobs in its queue in FIFO order. After a service completion, the processor sends $r$ siblings (task messages) to the $r$ voter queues in node $i+1$. If the processor is non-faulty, those siblings will agree with others produced by non-faulty processors in node $i$; other-
wise they will not.

There are two exceptions to the model in figure 2, namely nodes 1 and \( N \). In node 1, there are no voters and voter queues; jobs coming into the system from the outside are replicated on arrival into \( r \) siblings which immediately join the \( r \) service queues there. On the other hand, the service processors in node \( N \) produce a single result after each service completion (rather than \( r \)). There is a single final voter, with its queue, which arbitrates over the output of node \( N \).

Jobs arrive into the system in a Poisson stream with rate \( \alpha \). Service times at the voters and service processors in node \( i \) are exponentially distributed with means \( 1/\mu_i \) (\( \mu_i \) is the average voting rate) and \( 1/s_i \) (\( s_i \) is the average service rate), respectively (the Poisson and exponential assumptions are of course not necessary when the system is simulated, but are needed for the analysis). The average transit time between nodes \( i \) and \( i+1 \) (for \( i < N \)) is \( 1/\mu_i \); it is independent of how many messages are being transferred in parallel. The distributions of the transit times are immaterial (but see the simplifying approximations in section 4).

The system reliability aspects are modelled as follows. A processor fault (where the term 'processor' is interpreted in the sense of figure 1), is manifested by a fault in the corresponding service processor (in the sense of figure 2). Faults occur in different service processors - whether in the same node or in different ones - independently of each other. The intervals of non-faulty operation of processors in node \( i \), called 'up-times', are exponentially distributed with mean \( 1/\mu_i \) (\( \mu_i \) represents the average processor failure rate). Two patterns of faulty behaviour will be examined:

(i) Having once become faulty, a processor remains so until the end of the observation period.
(ii) A faulty processor in node $i$ is repaired, or is replaced by a new one, after a delay, called 'down-time', whose average length is $1/d_i$.

The models resulting from these two assumptions correspond to systems without reconfiguration and systems with reconfiguration; they will be referred to as 'model 0' and 'model 1', respectively. In both cases, the behaviour of a faulty processor differs from that of a non-faulty one only in that the former produces wrong results.

One immediate consequence of the above assumptions is that the condition for stability, i.e. for existence of steady-state, does not involve the up-time and down-time parameters. The transit parameters are not involved either, because there is no queueing for transmissions. Thus, the system is stable if

$$a < v_i \text{ and } a < s_i, \quad i = 1, 2, \ldots, N.$$  

These conditions are assumed to hold.

The performance measures in which we are interested are: (a) The average sojourn time, $W$ (interval between arrival into and departure from the system), for jobs that are completed successfully; and, (b) the distribution of the system operative state. The latter has a different interpretation in models 0 and 1, which will be clarified in section 4.

4. Analytical approximations

To represent completely the state of a system employing N-modular redundancy, one would have to specify not only the numbers of jobs in each service and voting queue, and in transit between nodes, but also the individual identities of the jobs in all waiting, service and transit positions. This is necessary in order to keep track of the siblings of any given job, so as to account for matching delays at the voters. A representation of this type is of
course possible. Given a suitable set of assumptions, it would lead to a vector-valued Markov process which could, in principle, be solved numerically. However, the size and complexity of the problem are such that, in practice, an exact solution is generally unattainable. Simulations, on the other hand, while perfectly feasible, tend to be very expensive in both computer utilisation and elapsed time.

It is desirable, therefore, to devise an approximate solution method whereby estimates of performance measures can be obtained cheaply and quickly, albeit with some loss of accuracy. This is our objective in the present section. Certain simplifying steps are taken in the approximation. The first of these is to assume that all siblings of a job arrive at the service queues in a node at the same time. This is of course true in node 1, but not necessarily in subsequent nodes. However, since the voters act as synchronisation points for siblings, and since voting times and transit times between nodes are usually small compared to the service times, the assumption is not unreasonable. In fact, we shall see that even when voting and transit times are not small, the accuracy of the approximation is acceptable.

The second simplifying step concerns the distribution of the interval between the arrival of a sibling at a service queue, and its arrival at the following voter. That interval, which includes waiting, service and transit (excepting the latter in the case of node N), will be referred to as the 'passage time' for the given node. Now, it is well known (e.g., see [Mitra87]) that if jobs arrive into a single-server queue in a Poisson stream with rate $\lambda$, and have exponentially distributed service times with mean $1/\mu$, then in the steady-state their response times are exponentially distributed with mean $1/(\mu - \lambda)$. We shall assume that the addition of the subsequent transit time does not destroy that exponentiality (which it does, in general). The
passage times for node $i$ will be treated as exponentially distributed random variables with mean

$$P_i = \frac{1}{s_i} + \frac{1}{t_i},$$

where $s_i$ is the service rate at node $i$, $a$ is the (common) job arrival rate and $1/t_i$ is the average transit time from node $i$ to node $i+1$. Note that $1/t_N=0$ by definition.

Suppose that the degree of replication in a node is $r$, so that $m$ correct results are necessary for a majority, where $m=\lfloor r/2 \rfloor$. Then, assuming that at least $m$ of the processors in the node are operating correctly, a following voter will be able to carry out a successful voting on a job when the first $m$ of the job's correctly executed siblings complete their passage times through the node. To estimate the average time until the occurrence of that event, we shall use the following known result:

Let $X_1, X_2, \ldots, X_n$ be a sample of $n$ i.i.d. (independent and identically distributed) random variables distributed exponentially with parameter $\mu$. Consider the order statistics of that sample, $Y_1, Y_2, \ldots, Y_n$. In other words, $Y_1$ is the smallest of the $X$'s, $Y_2$ is the second smallest, etc. Then the expectation of $Y_k$ is given by $E(Y_k) = (1/\mu)H_{n,k}$, where

$$H_{n,k} = \sum_{j=0}^{k-1} \frac{1}{n-j}, \quad k=1, 2, \ldots, n.$$  

Suppose that node $i$ has $c$ operative processors $(c>m)$. Denote by $w_i(c)$ the average associated delay time, i.e. the average period between the arrival of a job's siblings at the service queues of node $i$ and their arrival at the service queues of node $i+1$ (or the departure of the job from the system if $i=N$). Since the delay time consists of the $m$'th smallest passage time of the job's correctly executed siblings, plus the subsequent queueing and voting time, we can use the approximation
\[ w_i(c) = P_i H_{c,m} + \frac{1}{u_{i+1} - a}, \quad (3.3) \]

where \( P_i \) and \( H_{c,m} \) are given by (3.1) and (3.2) respectively, and \( u_{i+1} \) is the service rate of the voters in node \( i+1 \) (or of the final voter, if \( i=N \)).

N. B. In the case of a TMR node with all three processors operating correctly, we have \( m=2 \) and \( H_{3,2}=5/6 \), so that the first term in (3.3) is actually smaller than the corresponding average passage time in an unreplicated system. If voting times are small compared to service and/or transit times, then the TMR system will perform better than the unreplicated one. This is a reflection of the fact which (3.2) quantifies for the exponential distribution, namely that the second best out of three realisations of a random variable tends to be better than a single realisation. On the other hand, if only two out of the three processors are operative, then \( H_{2,3}=3/2 \), and the replicated system has a worse performance that the unreplicated one. These phenomena will be illustrated by the experimental results in section 4.

Thus, the average delay time associated with a replicated node depends on the operative state of that node, i.e. on how many of its processors produce correct results. The next step in the approximation is to estimate the probability distribution of the operative states. In order to simplify the state description, by not having to identify each node, and to avoid the proliferation of parameters, we shall assume from now on that all nodes are statistically identical. In particular, \( s_i=s, t_i=t \) and \( u_i=v \), for all \( i \) (but \( 1/t_N=0 \)). Moreover, we shall concentrate on the case of a TMR system, i.e. \( r=3, m=2 \). However, it should be clear that our methods apply to more general systems.

A node is said to be 'fully operative' if all three of its processors operate correctly. If only two of them do so, then the node is said to be 'partially operative'. The entire system is said to be 'operative' if every node is either fully or partially operative. Denote by \( q_j \) the conditional probability that
there are \( j \) fully operative nodes (and \( N-j \) partially operative ones), given
that the system is operative; \( j=0,1,\ldots,N \). If these probabilities are known,
then the average response time of a successfully completed job, \( W \), i.e. the
interval between the job's arrival into and departure from the system, can
be approximated by

\[
W = \sum_{j=0}^{N} q_j [jw(3) + (N-j)w(2)],
\]

(3.4)

where \( w(c) \) is given by (3.3), with \( s_i=s, u_i=v \) and \( t_i=N/(N-1) \) (to account for
the fact that \( 1/t_N=0 \)). Note, that (3.4) relies on equilibrium being reached
within each operative state of the system, i.e. on intervals between server
breakdowns being much larger than job interarrival, service, transit and
voting times. This is usually true in practice.

Our object now is to estimate the probabilities \( q_j \). Since the analysis is
different for models 0 and 1, the two cases will be considered separately.

4.1. Operative state distribution for model 0.

Recall that in model 0 all nodes start fully operative, and whenever a
processor breaks down, it remains broken for ever. The operative periods, or
'up-times', for individual processors are assumed to be i.i.d. random vari-
ables distributed exponentially with mean \( 1/u \). Consider the interval until
the first breakdown, i.e. the time during which all nodes are fully operative.
Call that interval the '0-period'. Since the 0-period starts with \( 3N \) operative
processors and ends when the first of them breaks down, its average length,
\( A_0 \), is, according to (3.2),

\[
A_0 = \frac{1}{3Nu}.
\]

(3.5)

Similarly, call the interval, if any, during which there are \( j \) partially
operative nodes and \( N-j \) fully operative ones, the '\( j \)-period'. Let \( A_j \) be its
average length. In order that the \( j \)-period exists, the processor breakdown
that terminates the \((j-1)\)-period must occur in one of the fully operative
nodes (the latter contain \(3(N-j+1)\) processors, out of a total of \(3N-j+1\)
operative ones in the system). If the \(j\)-period does exist, then its average
duration is \(1/(3N-j)u\). Hence we can write

\[
A_j = \frac{3(N-j+1)}{3N-j+1} \frac{1}{(3N-j)u}, j=1, 2, \ldots, N. \tag{3.6}
\]

The average interval during which the system is operative, \(A\), is of
course equal to

\[
A = \sum_{j=0}^{N} A_j. \tag{3.7}
\]

If we are interested in the behaviour of the system over a period of
time, \(T\), which is large compared to \(A\), then the conditional probability that
there are \(j\) fully operative nodes, given that the system is operative, can be
estimated as

\[
q_j = \frac{A_{N-j}}{A}, j=0, 1, \ldots, N. \tag{3.8}
\]

If, however, \(T\) is not large compared to \(A\), then an adjustment has to be
made to account for the possibility that \(T\) expires during one of the \(j\)-
periods. A simple way of making that adjustment is to define the quantities,
\(B_j\), as

\[
B_j = \min(\sum_{k=0}^{j} A_j, T), j=0, 1, \ldots, N. \tag{3.9}
\]

Then, the probabilities \(q_j\) can be estimated from

\[
q_{N-j} = \frac{B_j - B_{j-1}}{B_N}, j=0, 1, \ldots, N, \tag{3.10}
\]

where \(B_{-1}=0\) by definition.

4.2 Operative state distribution for model 1.

In model 1, when a processor breaks down, it is repaired (or replaced by
a new identical one), after a delay called the 'down-time'. Down-times are
i.i.d. random variables with mean $1/d$. In this model, neither the up-times nor the down-times need to be exponentially distributed. However, we still assume that equilibrium is reached between consecutive changes in the system operative state.

In the long run, any given processor is operative for a fraction of time, $a$, given by

$$a = \frac{1/u}{1/u + 1/d} = \frac{d}{u + d}. \quad (3.11)$$

Hence, the probability that a given node is fully operative, $p_0$, is given by $p_0 = a^3$. The probability that the node is partially operative, $p_1$, is $p_1 = 3a^2(1-a)$.

The probability that the system is operative, $q$, is obviously equal to

$$q = (p_0 + p_1)^N = a^{3N}(3 - 2a)^N. \quad (3.12)$$

Finally, the conditional probability that there are $j$ fully operative nodes and $N-j$ partially operative ones, given that the system is operative, is of the Binomial type:

$$q_j = \frac{1}{q} \binom{N}{j} p_0^j p_1^{N-j}, \quad j=0,1,\ldots,N. \quad (3.13)$$

It is perhaps worth emphasising again that these are long-run results. They are valid only when the observation period, $T$, is large compared to the up-times and down-times. For this model, there is no simple short-run approximation analogous to (3.10). To analyse the performance of the system in the short run, under assumptions of exponentially distributed up-times and down-times, one would have to study the transient behaviour of the Markov process $\{J_t; t \geq 0\}$, where $J_t = j$ if at time $t$ there are $j$ fully operative nodes and $N-j$ partially operative ones ($j=0,1,\ldots,N$). Another (absorbing) state, say '-1', would be added to represent an inoperative system. Then, for instance, the first passage time from state $N$ to state -1 would
correspond to the interval during which the system is operative. While not really difficult, such an analysis is outside the scope of the present paper.

Despite their simplicity and roughness, the approximations described here give quite accurate estimates of system performance measures. This will be illustrated in the following section.

5. Experimental results

Two groups of results are presented here. In the first group, all performance estimates are obtained by both simulations and analytical approximations. As well as observing the effect of various parameters on performance, the aim of these experiments is to assess the accuracy of the approximations. Both model 0 and model 1 are examined. In the second group, the emphasis is on comparisons between the reliability and performance of replicated and unreplicated systems.

An experiment normally involves fixing all parameters except one (usually the average voting time), and then approximating and simulating the simplex and the TMR systems for different values of that parameter. For model 0, a 'simulation' consists of 10 independent runs which differ only by the random number streams. In model 1, a single long run is made, divided into 10 portions with equal number of jobs completed in each portion. These samples of observations are used to obtain point estimates and confidence intervals for the average sojourn time, and point estimates for the fraction of time that the system is operative. The confidence intervals are not shown in the figures; their half-width is always less than 5% of the point estimate.

In the case of TMR systems, the sojourn time is approximated as described in section 4. The simplex system is of course a special case, with voting times equal to 0. In fact, the 'approximation' is then the exact
steady-state result for queues in tandem:

$$W = \frac{N}{s-a} + \frac{N-1}{t}. \quad (5.1)$$

The relative error of the approximation, expressed as a percentage, is denoted by $e$:

$$e = \frac{W' - W}{W} \times 100, \quad (5.2)$$

where $W'$ is the simulation estimate and $W$ is the approximated one.

For $r=1$ (simplex processing), $e=0$ indicates that the system has reached steady-state during the simulation run. When $e<0$, the analytical approximation overestimates the average response time; otherwise it underestimates it. In the following, unless specified, the value of $e$ will be for TMR systems.

The choice of system parameters in each experiment is influenced by the extended exponentiality assumption mentioned in the previous section: the passage time at any node is an exponentially distributed random variable with mean

$$P = \frac{1}{s-a} + \frac{1}{t}. \quad (5.3)$$

We choose parameters $1/s, 1/a, 1/t$ such that all cases of $1/(s-a)$ being greater than, equal to, and less than $1/t$ are considered.

5.1. Accuracy of analytical approximations

5.1.1. Results for model 0

In all simulation experiments in this model, $1/a$ was fixed at 2 and the simulation time at 2000. The maximum value of $1/a$ was 200000 and the minimum 10000, for which a pipeline of 5 processors was operative for 100% and 57.8% of simulation time respectively, and that of 5 TMR nodes was operative for 100% and 89.8% of simulation time respectively.
$1/s = 0.5; \ 1/a = 2.0; \ 1/t = 0.67; \ Simulation\ Time = 2000; \ N = 5; \ 1/u_1 = 200000; \ 1/u_2 = 10000.$

Figure 3. Sojourn Time Vs. Voting Time For u1 & u2 in Model-0.

$1/s = 0.5; \ 1/a = 2.0; \ 1/t = 0.67; \ Simulation\ Time = 2000; \ Uptime = 2000000; \ N = 10 \ & \ 3.$

Figure 4. Sojourn Time Vs. Voting Time For N = 10 and N = 3 in Model-0.
Considering the case $1/(s-a)>1/t$, two sets of experiments were run for parameters \( \{ 1/s=1, 1/t=1.25, N=5 \} \) (where $1/(s-a)=2$), and \( \{ 1/s=1.5, 1/t=0.5, N=5 \} \) (where $1/(s-a)=6$), with 9 values of $1/u$ in the above mentioned range. In each experiment, the simulations for $r=3$ were carried out for 6 values of mean voting times taken around the threshold voting time for which the TMR sojourn time exceeds the simplex sojourn time. In the first case, $e$ was found to be between -4% and +2% and in the second case, it was negative and greater than -11%. For the case $1/(s-a)<1/t$, we had similar sets of experiments for \( \{ 1/s=0.2, 1/t=1.33, N=5 \} \) (where $1/(s-a)=0.2$), and \( \{ 1/s=0.5, 1/t=1.33, N=5 \} \) (where $1/(s-a)=2/3$). In all the simulations, the relative error varied between 6.88% and 0.80%. Figure 3 illustrates the results of experiments for \( \{ 1/s=0.5, 1/t=0.67, N=5, 1/u=200000 \} \) and \( \{ 1/s=0.5, 1/t=0.67, N=5, 1/u=10000 \} \). In the first case $e$ was less than 8.25% and in the second, it was around 1.28%. For $r=1$, in both cases, $e$ was around 0.08% and in the figure, the estimated and simulation sojourn times are represented by one and the same line. Note that $1/(s-a)=1/t$ for the experiments whose results are shown in figure 3.

We also carried out experiments to study the accuracy of analytical approximations for different values of $N$. The experiments were carried out for those sets of parameters for which the magnitude of $e$ was worse when $N=5$. Figure 4 illustrates the results of experiments for \( \{ 1/s=0.5, 1/t=0.67, 1/t=200000, N=10 \} \) and \( 3 \). When $r=3$, for both $N=10$ and 3, $e$ was maximum at 9.2% for zero mean voting time and around 8% for other voting times, with $e$ decreasing as mean voting time increases.

The value of $e$ was between -9% and +8% in the experiments carried out for the following sets of parameters: \( \{ 1/s=1, 1/t=0.2, 1/u=12500, N=(10,5,3) \} \) (where $1/(s-a)=2$) and \( \{ 1/s=0.2, 1/t=1.33, 1/u=12500, N=(15,10,3) \} \)
(where $\frac{1}{(s-a)} \approx 0.2$). For the first set of parameters, $e$ was no worse than -9.9%, -5%, and +5% when $N$ was respectively 10, 5, and 3; in the second case, $e$ was found to be no worse than -1.29%, +1.67%, and +8% when $N$ respectively took the value of 15, 10, and 3.

In all simulation experiments carried out in model 0, the value of $e$ can be seen to be between -11% and +8%.

5.1.2. Results for model 1

For all simulation experiments in this model, $1/a$ was taken to be 2. The downtimes is exponentially distributed with mean $1/d$. The range of $1/d$ was chosen to be 1% to 10% of the average uptime $1/u$ which is fixed at 1000. A batch of 2000 jobs are successfully completed during each of the 10 portions of a simulation run.

The experiments were run for $1/d = 10$, 50, and 100 with other parameters being $\{1/s=0.5, 1/t=0.67, N=5\}$. (As in model 0, each experiment will involve at most 6 simulations for TMR processing with varying values of $1/u$). In all three experiments, the value of $e$ was found to be positive and less than 7.63%, 6.32%, and 4.5% respectively; it was maximum when $1/d = 10$. Therefore, we chose $1/d=10$, to run two experiments with $N=10$ and $N=3$, where $e$ was positive and less than 8%.

For $1/d = 10, 25, 50, 100$, and 200, we performed three sets of experiments for the parameters $\{1/s=1.5, N=5, 1/t=(0.5,1.0,1.5)\}$. (i.e., a total of 15 experiments each for a given $1/d$ and a given $1/t$ and each involving at most six simulations for TMR processing with various values of $1/u$). Among these experiments, the relative error was in the range (-9.5%, 3%). For these worst cases, we carried out experiments with value of $N$ changed to 10 and 3. The worst relative error in the former case was -13% and in the latter the errors were much smaller.
Model-1: $1/s = 1.5; 1/a = 2; 1/t = 0.5; \text{up-time} = 1000; \text{down-time} = 50; N = (5, 10)$

Figure 5. Sojourn Time Vs. Voting Time For $N = 5, 10$ in Model-1.

Model-1: $1/s = 1.5; 1/a = 2; 1/t = 0.5; \text{up-time} = 1000; \text{down-time} = 100; N = (10, 3)$.

Figure 6. Sojourn Time Vs. Voting Time For $N = 10, 3$ in Model-1.
Figures 5 and 6 represent the worst cases (for $N=10$) and moderate cases (for $N=3$ or $5$) of model 1 with parameters $\{1/s = 1.5, 1/t = 0.5, 1/d = 50\}$ and $\{1/s = 1.5, 1/t = 0.5, 1/d = 100\}$ respectively. In figure 5 for $N=10$ and in figure 6, it can be observed that the actual and estimated sojourn times for $r=1$ are significantly far apart. (Recall that the sojourn time estimates for unreplicated system are accurate only in steady state). This difference indicates that when $N$ becomes large for given $1/u$ and $1/d$ or when $1/d$ increases for a given value of $1/u$, the period during which the unreplicated system is in transient state (following a processor failure and replacement) becomes considerable.

To cover the case $1/(s-a)<1/t$, three experiments were carried out for $\{1/s = 0.2, 1/t = 1.33, N = 5\}$ with values of $1/d$ being 10, 50, and 100. In all the experiments, $\epsilon$ was positive and less than 5.35%.

To summarise this subsection: To study the accuracy of analytical approximation methods for evaluating the task sojourn times, a total of approximately 30 simulation experiments were performed for different sets of parameters in each model. Each experiment involved, for a given set of parameters, one simulation for $r=1$ and at most 6 simulations with different values of $1/u$ for $r=3$. The relative error was found to be between -11% and +8% in model 0 and between -13% and +8% in model 1.

5.2. Performance of unreplicated and TMR systems

For $N=5$, and 10, and simulation time 2000 in model 0, the operative periods of unreplicated and TMR systems as a % of simulation time for different uptimes are shown in figure 7. when $N=5$, the TMR system starts dropping from 100% operation for $1/u = 14000$ at which the unreplicated system is operative only for 65%. In figure 8, %operative times of TMR and unreplicated systems are shown for different values of $1/d$ with $1/u = 1000$
Model 0: $N = (5,10)$; Simulation Time = 2000.

Figure 7. (System operative period / Simulation time) in % Vs Uptime in model-0.

Model-1: $N = (5,10)$; uptime = 1000; Simulation period = completion of 20000 jobs.

Figure 8. % System operative time Vs. Downtime in Model-1.
and $N=5$, and 10. When $N=5$, the TMR system falls below 100% for a
downtime value around 13 at which the unreplicated system is operative for
around 93% of mission time.

For given $1/u$, simulation time, and $1/d$ (for model 1), as $N$ increases the
% operative time of both unreplicated and TMR systems decreases. This is
intuitively obvious, since there are more processors liable to fail. Moreover,
the rate of decrease is larger for the unreplicated than for the TMR system
(see figures 7 and 8).

The average sojourn time in a TMR system is lower than that of the
underlying system when voting times are low and all processors are
operative. This is due to the order statistics effect mentioned in section 4. As
voting times increase, or as processors faults become more frequent, the
TMR system starts performing worse. In figure 3, it can be seen that the
sojourn times for $1/u_2=10000$ are larger than those for $1/u_1=200000$ at
corresponding voting times. Figure 4 illustrates the increase in sojourn
times with increase in $N$, with all other parameters being the same: for
example, at $1/v=0$, the TMR sojourn times (by simulation) for $N=10$ and
$N=3$ are respectively 11.5 and 3 time units, or 1.15 and 1.0 time units per
node. From figures 5 and 6, increase of sojourn times with increase in $1/d$
can be observed (for example, for $N=10$ and $1/v=0.3$, the sojourn times (by
simulation) are 58.0 and 60.0 respectively in figure 5 (where $1/d=50$) and in
figure 6 (where $1/d=100$)). For all simulations carried out in model 0 with
$N$ varying from 10 to 3 and $1/u$ from 200000 to 10000, the TMR sojourn
time for $1/v=0$ was less than the corresponding unreplicated sojourn time.
Similar results were observed for all simulations carried out in model 1 with
$1/u=1000$ and $N$ varying from 10 to 3, except for the case of $1/d=200$.
When the TMR sojourn time at $1/v=0$ is less than the unreplicated sojourn
Model-0: \(1/a = 2; 1/t = 0.5; N = 5; \) uptime = 15000; simulation time = 2000.

![Graph 9](image)

Figure 9. Sojourn Time Vs. Service Time with voting time = 10% service time.

Model-1: \(1/a = 2; 1/t = 0.5; N = 5; \) Uptime = 1000; Downtime = 50.

![Graph 10](image)

Figure 10. Sojourn Time Vs. Service Time with voting time = 10% of service time.
time, there exists a threshold average voting time at which the TMR sojourn time equals the unreplicated sojourn time. In a system, where TMR nodes are likely to be fully operative, the threshold value will increase with an increase in mean passage time; when TMR nodes are likely to be partially operative, it decreases with an increase in mean passage time. Taking \(1/u\) to be 10\% of \(1/s\), figures 9 and 10 show TMR and unreplicated sojourn times for various values of \(1/s\) respectively with parameters \(\{1/a=2, 1/t=0.5, N=5, 1/u=15000, \text{ simulation time } 2000\}\) for model 0 and with parameters \(\{1/a=2, 1/t=0.5, N=5, 1/u=1000, 1/d=50\}\) for model 1. For the chosen parameters, the actual and estimated sojourn times for TMR system are less than those times for unreplicated system in both the models.

6. Concluding remarks

A general model of distributed replicated processing was presented in section 2. The model captures essential details of practical systems reported in the literature (e.g. [Wensl78, Powel88]). The factors capable of affecting the performance of NMR system in relation to the simplex system were identified to be (i) voting times, (ii) processor failure rates, (iii) extra message traffic, and (iv) sequencing overheads. In this study, we have assumed a specialised pipeline architecture with high bandwidth communication which enables us to ignore the effects of extra message traffic and of sequencing overheads. The model of such a system was presented in section 3. The performance of such a system was then evaluated both by computer simulations and by analytical approximations. Despite their simplicity and roughness, the approximations developed here have been shown to estimate the system performance measures fairly accurately - thus providing an attractive alternative to time consuming simulation experiments.
We have compared the mean sojourn times of jobs in replicated and unreplicated systems. A rather surprising result has been that when voting times are small and processor failures are less likely, a replicated system can provide better response times. This is because when redundant processors in each node are considered to be unevenly loaded, the presence of extra processors brings performance benefits by exploiting the earliest service completions of siblings. We have also compared the improvement in system operative time obtained as a result of replicating processors in systems with and without reconfiguration. In particular, we see that when the reconfiguration time (downtime) is a small percentage of the uptime of a processor, the replicated system can provide continuous service. These and other performance measures have been discussed in detail in section 5.

The particular assumptions that we have made are not the only ones that can be handled by our approximation approach. For example, rather than assigning the voting and computational functions to separate processors, one could use a single processor with appropriately modified service times. Also, the assumption that congestion has no effect on transit times can be relaxed. This and other generalisations of the models would provide a relevant topic of further research.

7. Acknowledgements

This work has been supported in part by a grant from the U.K. Science and Engineering Research Council.
References

Lampo82.

Manci86.

Mitra87.

Pitte86.

Powel88.

Shriv87.

Wensi78.

York83.