Approximate Availability Analysis of VAXcluster Systems

Oliver C. Ibe, Member IEEE
Digital Equipment Corp., Andover
Richard C. Howe
Digital Equipment Corp., Andover
Kishor S. Trivedi, Senior Member IEEE
Duke University, Durham

Key Words **— Hierarchial modeling, Availability analysis, Distributed system, VAXcluster**

Reader Aids —
Purpose: To report a new modeling technique
Special math needed for explanations: Probability
Special math needed to use results: None
Results useful to: Reliability analysts, System analysts

Abstract **— We solve for the availability of an *n*-processor VAXcluster system using a hierarchical approach that allows us to: 1) obtain a closed-form answer to an apparently difficult problem, and 2) determine the optimal number of processors in the cluster for a given set of cluster parameters. Our novel approach is a 2-level hierarchical model in which the lower-level model is a 9-state Markov chain that is solved in a closed form. The 9-state Markov chain is then aggregated into a 3-state device analogous to a *diode*. Subsequently, the system availability is computed by analyzing a simple network.**

1. INTRODUCTION

A VAXcluster is a closely-coupled multicomputer system with two or more VAX processors, one or more mass storage servers called Hierarchical Storage Controllers (HSCs), a set of disks, and a star coupler (SC) [1]. A VAXcluster with two processors, two HSCs, and two disks is shown in figure 1. From an availability point of view, a VAXcluster is essentially a "series-parallel" system consisting of a "parallel" network of *n* processors in "series" with a "parallel" network of HSCs and a "parallel" network of disks [2]. Figure 2 is the reliability block diagram of the VAXcluster in figure 1. The star coupler is assumed to be a passive connector and so is extremely reliable. Therefore, it is omitted in the block diagram. The block diagram partitions the VAXcluster along functional lines; this permits us to model each component type separately. We consider only the processor subsystem. The terms VAXcluster and cluster are used interchangeably.

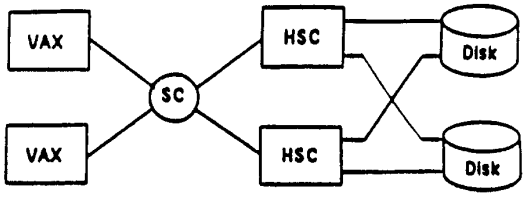

Fig. 1. Physical Configuration of a Two-Processor VAXcluster

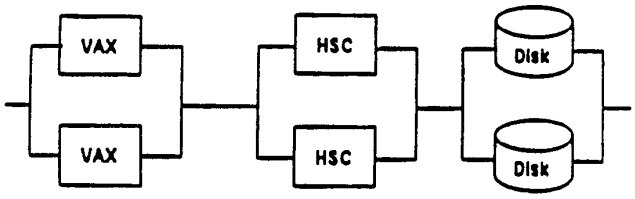

Fig. 2. Reliability Block Diagram of the Two-Processor Cluster

We assume that there are several processors in the system. Each processor can suffer one of two types of failures: *permanent* and *intermittent*. A permanent failure requires the physical repair of the failed processor. By contrast, after a processor has suffered an intermittent failure, no physical repair is required. The processor is restored to operation by means of a processor reboot. Each of these failure types is a cluster fault. A cluster fault that arises from a permanent failure of any of the processors is a permanent fault, and a cluster fault that arises from an intermittent failure of any of the processors is an intermittent fault.

A cluster fault can be *covered* or *uncovered*. Technically, the distinction beween a covered cluster fault and an uncovered cluster fault lies in the time it takes for the other operational (or up) processors in the cluster to resume operation after a fault. After a covered fault, the system undergoes a *cluster transition* and the up processors continue the processing. A cluster transition takes on the order of seconds to complete. Thus if a quorum (minimum number of processors required for a VAXcluster to function) is still formed by the up processors, a covered fault causes a small loss in system time. An uncovered fault is one that causes the cluster to go down. This then requires the cluster to be rebooted before the up processors can continue their processing, given that they still form a quorum. Thus, recovery from an uncovered fault, whether permanent or intermittent, takes more time than that from a covered fault.

The availability of an *n*-processor VAXcluster system can be obtained by means the continuous-time Markov chain

0018-9529/89/0400-0146$01.00©1989 IEEE

chain if we assume that the times between failures, the repair times and other recovery times are exponentially distributed. The restriction of exponential distribution can often be relaxed by resorting to the method of stages [3, 4]. However, even for small values of n the state space of the Markov chain can be very large. Our experience is that when $n > 5$, the system is difficult to solve by such software packages as SHARPE [5] which are widely used in availability analysis. Even if the packages were able to solve such large Markov models, the crucial question of generation of such models needs to be addressed.

We consider a decomposition method for computing availability for a VAXcluster with n processors. The basic approach is:

1. We construct the Markov chain of the behavior of a single processor.

2. We combine these individual Markov chains in a hierarchical manner to obtain the approximate behavior of the system.

The problem thus becomes a combinatorial one that lends itself to extensions to any number of processors. □

The major contributions of this work include:

1. We reduce the complexity of a large system by the process of decomposition.

2. We show that for a given set of processor parameters there is an optimal n that maximizes the system availability. Under some mild assumptions, the optimal n has a closed-form solution. In the more general case the optimal n can be computed numerically. □

In a companion paper [6] we also use the 2-level hierarchical model, where the bottom level is a Markov chain and the top level is a combinatorial model. The interesting feature of this paper, however, is that the top-level is a network of *diodes* (3-state devices) rather than a conventional combinatorial model with 2-state devices.

The terms, "series" & "parallel" are used in their logic sense, not in their layout or schematic diagram sense; they are shown in quotes for that reason.

Notation

$1/\lambda_P$	mean time between permanent faults
$1/\lambda_I$	mean time between intermittent faults
$1/\mu_P$	mean time to repair a processor (that suffers a permanent fault)
$1/\mu_{PB}$	mean time to reboot a processor
$1/\mu_{SB}$	mean time to reboot a cluster
$1/\mu_T$	mean duration of a cluster transition
c	coverage factor for permanent faults
k	coverage factor for intermittent faults
A_i	availability of an i-processor cluster, $i = 1,\ldots,n$
P_l	steady-state probability that the system is in state l

Other standard notation is given in "Information for Readers & Authors"at the rear of each issue.

2. THE BASIC MODEL

1. There is an n-processor VAXcluster, where $n \geq 2$. Each processor is either up or down.

2. The time to the occurrence of a permanent fault, given that a processor is up, is exponentially distributed with mean $1/\lambda_P$.

3. Given that a processor is up, the time to the occurrence of an intermittent fault is exponentially distributed with mean $1/\lambda_I$.

4. The c is the probability that a permanent fault is covered and $1 - c$ is the probability that it is an uncovered fault. An analogous definition can be given for k.

5. The faults occurring in one processor are statistically independent of those in another processor.

6. When a covered fault occurs in any of the processors, the up processors go through a *cluster transition.* This means that these processors temporarily suspend operations, cancel the cluster membership of the down processor, check to see if the up processors still form a quorum, and recommence their processing if a quorum is formed. (If a quorum is not formed, no processing is done and the cluster is defined to be down.) The duration of a cluster transition is exponentially distributed with mean $1/\lambda_T$.

7. The time to repair a processor that suffers a permanent fault is exponentially distributed with mean $1/\mu_P$.

8. The time to reboot a processor (which suffers a covered intermittent fault or an uncovered intermittent fault, if it is the only up processor before it failed) is exponentially distributed with mean $1/\mu_{PB}$.

9. After a processor with a permanent fault has been repaired, or a processor with a covered intermittent fault has been rebooted, the system goes through another cluster transition which consists of readmitting the processor into the cluster. When an uncovered fault occurs, the cluster goes down and needs to be rebooted. After the reboot, the up processors continue processing if they form a quorum. Then the repair of the down processor commences if the fault is permanent, or the processor reboot commences if the fault is intermittent. The time to reboot the cluster is exponentially distributed with mean $1/\mu_{SB}$.

10. The cluster transition time is so small that no other event (such as processor failure or repair completion) can take place during a cluster transition.

Thus, we can model the n-processor VAXcluster by a continuous-time Markov chain. Figure 3 shows the model for $n = 2$, and figure 4 shows the model for $n = 3$.

Figure 3 can be explained as follows. In state 1 the two processors are up. If either of them suffers a permanent failure that results in a covered permanent fault, the system enters state 2 from where the cluster undergoes a transition. At the end of the cluster transition, the system enters state 3. In this state the cluster is up with only one

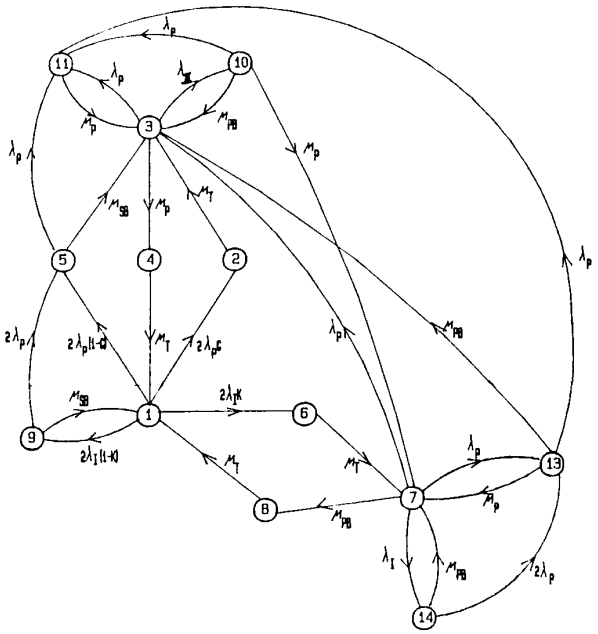

Fig. 3. Markov Chain for a 2-Processor VAXcluster

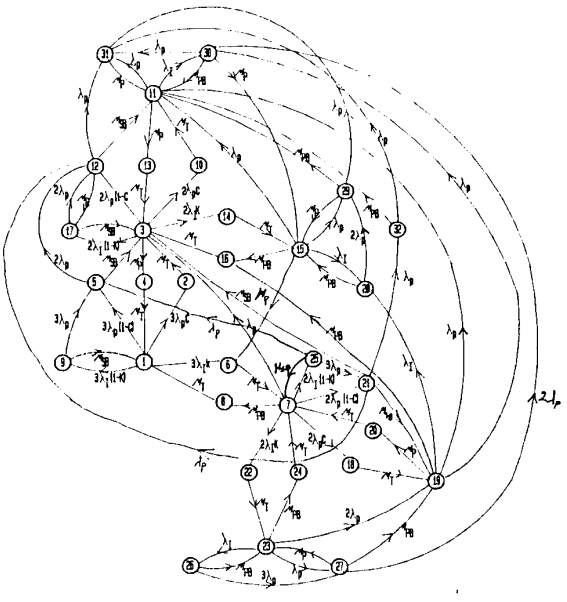

Fig. 4. Markov Chain for a 3-Processor VAXcluster

processor. When the failed processor is repaired, the system enters state 4 where another cluster transition takes place. The system returns to state 1. Given that the system is in state 1, the first uncovered permanent fault brings the system to state 5. The cluster is then rebooted. At the end of the cluster reboot the system enters state 3. While the cluster is being rebooted (in state 5), the up processor can suffer a permanent failure. This will cause the system to enter state 11 where the cluster is down due to both processors suffering permanent failures. Another way the system can enter state 11 is if it was in state 3 and the up processor suffers a permanent failure.

Given that the system is in state 3, it will enter state 10 if the up processor suffers an intermittent failure. While the system is in state 10 one of three events can take place. If the repair of the processor that suffered a permanent failure is completed before the processor that suffered an intermittent failure is rebooted, then the system enters state 7. Otherwise, it returns to state 3. The processor that suffered an intermittent failure is not immune to a permanent failure since it is powered up while the reboot is taking place. Thus given that the system is in state 10, if the processor being rebooted suffers a permanent failure before the reboot is completed, the system enters state 11 where, as we noted earlier, the system is down.

Given that the system is in state 1, a covered intermittent fault will cause the system to enter state 6. The cluster then undergoes a transition, and the system enters state 7. In state 7 one processor is up and one is down for reboot. When the reboot is completed, the system enters state 8 from where it returns to state 1 after another cluster transition. While in state 7 one of three events can take place. If the processor being rebooted suffers a permanent failure, the system enters state 3. Such a processor failure does not affect the operation of the cluster since the processor being rebooted has already been mapped out of the cluster. If the up processor suffers a permanent failure, then the system enters state 13 from where the completion of the reboot takes the system to state 3. The completion of the repair returns the system to state 7. While in state 13, if the processor being rebooted suffers a permanent failure, then the system enters state 11. Finally, if the up processor suffers an intermittent failure, the system enters state 14 from where the completion of a reboot takes it back to state 7. While in state 14 either of the two processors being rebooted can suffer a permanent failure which causes the system to enter state 13.

Given that the system is in state 1, the first uncovered intermittent fault causes the system to enter state 9. This calls for a cluster reboot the completion of which returns the system to state 1. While in state 9, either of the two processors can suffer a permanent failure which takes the system to state 5. The repair of the processor will commence in state 3 at the completion of the cluster reboot.

The availability of the 2-processor VAXcluster, which is the steady-state probability that at least one processor in the cluster is up, is:

$$A_2 = P_1 + P_3 + P_7 \quad (1)$$

This definition of availability does not consider the computational capacity available to the user in each of the 3 up states. One way to incorporate the available computational capacity in each up state is to consider a Markov reward model [7] that weights an up state according to the computational capacity of the cluster when it is in the state. This leads to an availability measure we refer to as *performance-oriented* availability. For example, if the computational capacity of the cluster when it is in states 3 and 7 is one half that when it is in state 1, then we can apply the weight 1 to state 1 and weight 0.5 to states 3 and 7. Thus, the performance-oriented availability is:

$$A_p = P_1 + 0.5(P_3 + P_7)$$

In the remainder of the paper we restrict our discussion to the *traditional* availability as defined in (1).

The state transitions of the 3-processor VAXcluster in figure 4 can be explained in a similar manner. State 1 is the initial state where all 3 processors are up. In state 3, the cluster is up on 2 processors after one processor that suffered a permanent fault has been mapped out of the system. In state 7, the cluster is also up on 2 processors after a processor which suffered an intermittent failure that caused a covered intermittent fault has been mapped out of the cluster. In state 11 the cluster is up on only one processor with 2 processors down with permanent failures. In states 15 and 19, the cluster is up on one processor; one processor is down with a permanent failure and the other is down with an intermittent failure that resulted in a covered intermittent fault. Finally, in state 23 the cluster is up on one processor; the other 2 processors are down with intermittent failures that resulted in covered intermittent faults. Since at least one processor must be up for the cluster to be up, the availability of the 3-processor VAXcluster is:

$$A_3 = P_1 + P_3 + P_7 + P_{11} + P_{15} + P_{19} + P_{23}. \quad (2)$$

Figures 3 and 4 show that the state space of the Markov chain for an n-processor VAXcluster increases exponentially with n, thereby making the availability difficult to analyze at large values of n. In the next section we develop an approximate analysis for an n-processor VAXcluster which can be used for any value of n. The analysis is based on the following observation. From the definition of a covered fault, a processor fault that breaks the quorum cannot be covered. Therefore, if we consider the behavior of individual processors, then a processor fault that does not break the quorum can be modeled as shown in figure 5, and a processor fault that breaks the quorum can be modeled as shown in figure 6.

3. A NEW DECOMPOSITION TECHNIQUE

To obtain an approxiate analysis of the system availability, we make the following assumptions.

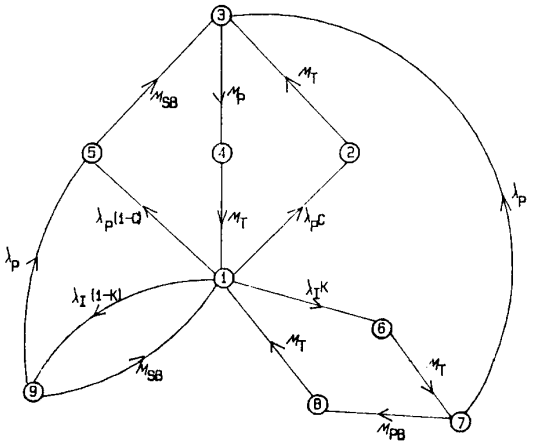

Fig. 5. Model of a Processor Fault That Does Not Break the Quorum

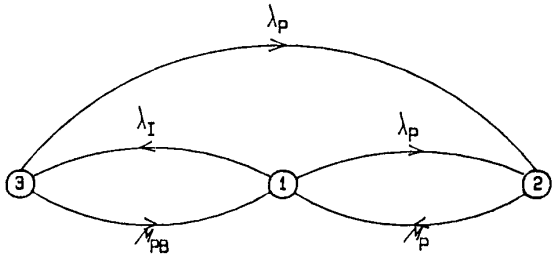

Fig. 6. Model of a Processor Fault That Breaks the Quorum Shown in Figure 5

1. Every processor is modeled as if it is not the one to break the quorum. Thus all processors are modeled by the Markov chain shown in figure 5.

2. Each processor has a statistically independent repairman.

Assumption 1 is justified if the probability that the cluster goes down due to loss of quorum [8] is relatively low. Assumption 2 is justified if the MTBF (mean time between failures) is large compared to the MTTR (mean time to repair) so that the time a faulty processor spends waiting for the repair crew to arrive is negligible. These two assumptions allow us to decompose the system into n statistically independent subsystems.

With respect to figure 5, we define the *superstates*:

X set of states in which the processor is up = {1}

Y set of states in which the cluster is down due to an interference from the processor = {2, 4, 5, 6, 8, 9}

Z set of states in which the processor is down but the cluster is up, given that a quorum is formed = {3, 7}

More Notation

P_k probability that a processor is in superstate k, $k \in \{X, Y, Z\}$.

n_X, n_Y, n_Z number of processors in superstate X, Y, Z, respectively.

The availability, A, of the VAXcluster is: $A \equiv \Pr\{$at least one processor in superstate X and none in superstate $Y\}$. The availability A_n of an n-processor VAXcluster is:

$$\begin{aligned} A_n &= \sum_{n_X=1}^{n} \binom{n}{n_X\, 0\, n - n_X} P_X^{n_X} P_Y^0 P_Z^{n-n_X} \\ &= \sum_{n_X=1}^{n} \frac{n!}{n_X!(n-n_X)!} P_X^{n_X} P_Z^{n-n_X} \\ &= \sum_{n_X=0}^{n} \binom{n}{n_X} P_X^{n_X} P_Z^{n-n_X} - \frac{n!}{0!(n-0)!} P_X^0 P_Z^n \\ &= (P_X + P_Z)^n - P_Z^n \end{aligned} \tag{3}$$

To optimize over n we use $dA_n/dn = 0$; ie:

$$P_Z^n \ln(P_Z) - (P_X + P_Z)^n \ln(P_X + P_Z) = 0$$

And the optimal n is:

$$n^* = \frac{\ln[\ln(P_Z)/\ln(P_X + P_Z)]}{\ln(P_X + P_Z) - \ln(P_Z)} \tag{4}$$

The above model is similar to a "parallel" network of *diodes* [9, 10] as shown in figure 7. A *diode* is a 3-state device that can be in one up state and two down states: shorted and open circuit. In a "parallel" network of n *diodes*, when any *diode* is shorted, the system fails. An open circuit does not cause the system to fail if at least one *diode* is functioning properly. Thus, the availability of the "parallel" networks of *diodes* is the probability that no *diode* is short and at least one *diode* is functioning properly. That is, the short circuit corresponds to the superstate Y in our model, the open circuit corresponds to our superstate Z, and the up state corresponds to the superstate X.

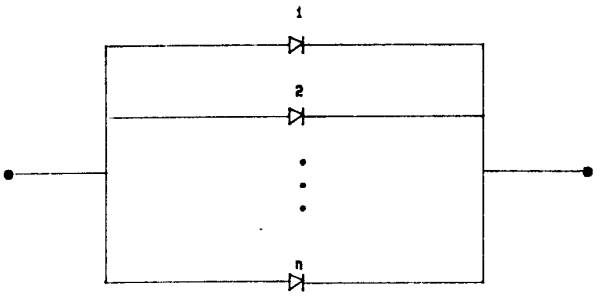

Fig. 7. A Parallel Network of Diodes

3.1 An l-out-of-n Operation

Generally the VAXcluster operates in the l-out-of-n mode in the sense that when a fault occurs, the up processors must form a quorum if the system is to recover from the fault [8]. When we require that at least l out of n processors be up, the availability is:

$A_{l/n} \equiv \Pr\{$at least l processors in superstate X and none in superstate $Y\}$

$$\begin{aligned} &= \sum_{n_X=1}^{n} \binom{n}{n_X\, 0\, n - n_X} P_X^{n_X} P_Y^0 P_Z^{n-n_X} \\ &= \sum_{n_X=0}^{n} \binom{n}{n_X} P_X^{n_X} P_Z^{n-n_X} - \sum_{n_X=0}^{l-1} \binom{n}{n_X} P_X^{n_X} \quad P_X^{n-n_X} \\ &= (P_X + P_Z)^n - \sum_{n_X=0}^{l-1} \binom{n}{n_X} P_X^{n_X} P_Z^{n-n_X} \end{aligned} \tag{5}$$

When $l = 2$, we obtain:

$$A_{2/n} = (P_X + P_Z)^n - nP_X P_Z^{n-1} - P_Z^n \tag{6}$$

with the optimizing n given numerically by the equation:

$$\begin{aligned} &P_Z^{n-1}[P_X + (nP_X + P_Z)ln(P_Z)] \\ &\quad = (P_X + P_Z)^n ln(P_X + P_Z). \end{aligned} \tag{7}$$

And when $l = n - 1$, we obtain:

$$A_{(n-1)/n} = nP_X^{n-1} P_Z + P_X^n \tag{8}$$

with the optimizing n satisfying the equation:

$$P_X^{n-1} P_Z + nP_X^{n-1} P_Z \ln(P_X) + P^{nX} \ln(P_X) = 0. \tag{9}$$

The values of P_X and P_Z can be obtained by solving the steady-state equations for the single processor model of figure 5. By using the ordinary balance equations we obtain [11]:

$$P_2 = \frac{\lambda_P c}{\mu_T} P_1 = a_2 P_1$$

$$P_3 = \frac{1}{\mu_P} \left\{ \lambda_P + \frac{\lambda_I \lambda_P k}{\lambda_P + \mu_{PB}} + \frac{\lambda_I \lambda_P (1-k)}{\lambda_P + \mu_{SB}} \right\} P_1 = a_3 P_1$$

$$P_4 = \frac{1}{\mu_T} \left\{ \lambda_P + \frac{\lambda_I \lambda_P k}{\lambda_P + \mu_{PB}} + \frac{\lambda_I \lambda_P (1-k)}{\lambda_P + \mu_{SB}} \right\} P_1 = a_4 P_1$$

$$P_5 = \frac{1}{\mu_{SB}} \left\{ \lambda_P (1-c) + \frac{\lambda_I \lambda_P (1-k)}{\lambda_P + \mu_{SB}} \right\} P_1 = a_5 P_1$$

$$P_6 = \frac{\lambda_I k}{\mu_T} P_1 = a_6 P_1$$

$$P_7 = \frac{\lambda_I k}{\lambda_P + \mu_{PB}} P_1 = a_7 P_1$$

$$P_8 = \frac{\lambda_I \mu_{PB} k}{\mu_T(\lambda_P + \mu_{PB})} P_1 = a_8 P_1$$

$$P_9 = \frac{\lambda_I(1 - k)}{\lambda_p + \mu_{SB}} P_1 = a_9 P_1$$

$$P_1 = \frac{1}{1 + \Sigma_{i=2}^{9} a_i} \tag{10}$$

Then—

$$P_X = P_1 \tag{11}$$

$$P_Z = (a_3 + a_7)P_X \tag{12}$$

$$P_Y = 1 - P_X - P_Z \tag{13}$$

4. Numerical Results

The following parameters and results are illustrative only; they do not represent any current or planned products.

Example 1.

$1/\lambda_P$ = 5000 hours, $1/\lambda_I$ = 2000 hours.
$1/\mu_P$ = 2 hours, $1/\mu_T$ = 30 seconds.
$1/\mu_{PB}$ = 6 minutes, $1/\mu_{SB}$ = 10 minutes.

Then solving the 9-state Markov chain in figure 5, we obtain the P_X, P_Y, P_Z shown in table 1 for various values of coverage factors. The exact models were solved by SHARPE. Table 2 shows the availability predictions of both the exact and approximate models for various coverage factors when $n = 2$. Table 3 shows the case when $n = 3$ and we require one out of the three processors to be up for the cluster to be up. The difference in the mean downtime per year of the two models is 3.2 seconds when $n = 2$, and 12.6 seconds when $n = 3$, where the mean downtime per year is computed as:
Mean downtime per year = (1 − Availability) × 365 × 24 × 60 minutes.

In this example, the approximate analysis gives results that are very close to the exact analysis.

TABLE 1
Values of P_k

c	k	P_X	P_Y	P_Z
0.90	0.90	0.9995329	0.0000223	0.0004448
0.90	0.94	0.9995339	0.0000193	0.0004468
0.94	0.94	0.9995351	0.0000181	0.0004468

TABLE 2
Availability Predictions at $n = 2$

c	k	A_{exact}	A_{apprx}
0.90	0.90	0.9999550	0.9999551
0.90	0.94	0.9999610	0.9999611
0.94	0.90	0.9999576	0.9999577
0.94	0.94	0.9999636	0.9999637

TABLE 3
Availability Predictions at $n = 3$

c	k	A_{exact}	A_{apprx}
0.90	0.90	0.9999326	0.9999330
0.90	0.94	0.9999416	0.9999420
0.94	0.90	0.9999364	0.9999368
0.94	0.94	0.9999454	0.9999458

□

Example 2.

$1/\lambda_P$ = 7000 hours, $1/\lambda_I$ = 4000 hours.
$1/\mu_P$ = 2 hours, $1/\mu_{PB}$ = 3 minutes.
$1/\mu_{SB}$ = 5 minutes, and $1/\mu_T$ = 15 seconds.

Table 4 shows the availability predictions of both the approximate and exact methods when $n = 2$, and table 5 shows the case $n = 3$. From the tables, when $n = 2$ the two models give identical results. When $n = 3$, the difference in mean downtime per year is only 6 seconds. Thus, under this set of parameter values, the approximate method gives results that are very close to the exact ones.

TABLE 4
Availability Predictions when $n = 2$

c	k	A_{exact}	A_{apprx}
0.90	0.90	0.9999873	0.9999873
0.90	0.94	0.9999888	0.9999888
0.94	0.90	0.9999882	0.9999882
0.94	0.94	0.9999897	0.9999897

TABLE 5
Availability Predictions when $n = 3$

c	k	A_{exact}	A_{apprx}
0.90	0.90	0.9999809	0.9999811
0.90	0.94	0.9999832	0.9999834
0.94	0.90	0.9999823	0.9999825
0.94	0.94	0.9999845	0.9999847

□

A more extensive validation of the approximate analysis is reported in [12]. The report, which considers up to $n = 5$, indicates that with the 1-out-of-n operating rule there is no appreciable difference between the exact and the approximate methods. (The differences in the results are similar to those reported here.) Slight differences in availability (generally less than one minute difference in mean downtime per year) are obtained when the cluster operates under the $(n - 1)$-out-of-n quorum rule, the so-called $n + 1$ *redundancy rule.* This rule defines the cluster to be down when two or more processors are down. For any quorum rule that is less stringent than the $n + 1$ redundancy, the approximate and exact availability values are identical within 1×10^{-8}. This assumes parameter values on the same order as those used in this paper.

. SUMMARY & DISCUSSION

We have developed an approximation method for computing the availability of the n-processor VAXcluster-system. The technique also provides information on the optimal number of processors in a VAXcluster that are required for a given set of processor parameters. The approximation on the independent repair crew is valid if the MTBF is small compared to the MTTR (the repair is fast) so that the delay in waiting for the repair crew is negligible. The other assumption (the last processor to experience a fault is treated in the same way as any other processor) means that the system does not have sequence-dependent behavior. As the results for the 2-processor and 3-processor VAXcluster indicate, these two assumptions do not cause any appreciable difference between the exact result and the approximate result.

The number of states in the exact Markov chain for an n-processor VAXcluster is $O(n^3)$ [12]. The computational complexitiy of the exact analysis depends on the solution method used. For instance, if a full storage iterative method is used, then the time complexity will be $O(n^6)$. The computational complexity of the approximate method is a constant for all n. Therefore, the approximate method results in considerably reduced savings in both storage space and computation time, especially at high values of n. A major issue in the VAXcluster availability analysis is the generation of all the states for a given value of n. This requires a great deal of time and effort. By using the approximate method we avoid this problem without losing much accuracy.

REFERENCES

[1] N. P. Kronenberg, H. M. Levy, W. D. Strecker, "VAXclusters: A closely-coupled distributed system", *ACM Trans. Computer Systems,* vol 4, 1986 May, pp 130-146.

[2] E. E. Balkovich, P. Bhabhalia, W. R. Dunnington,. T. F. Weyant, "VAXcluster availability modeling", *Digital Technical Journal,* 1987 Sep., pp 69-79.

[3] D. R. Cox, "A use of complex probabilities in the theory of stochastic processes", *Proc. Cambridge Philosophical Society,* vol 51, 1955, pp 313-319.

[4] M. C. Hsueh, R. K. Iyer, K. S. Trivedi, "Performability modeling based on real data: a case study", *IEEE Trans. Computers,* vol 37, 1988 Apr., pp 478-484.

[5] R. Sahner, K. S. Trivedi, *SHARPE: An Introduction and Guide to Users,* Duke University, Durham, NC 27706 USA. 1986.

[6] J. T. Blake, K. S. Trivedi, "Reliability analysis of interconnection networks using hierarchicial composition", *IEEE Trans. Reliability,* vol 32, 1989 Apr, pp 111-120..

[7] R. M. Smith, K. S. Trivedi, A. V. Ramesh, "Performability analysis: measures, an algorithm, and a case study", *IEEE Trans. Computers,* vol 37, 1988 April, pp 406-417.

[8] Digital Equipment Corporation, *VAXcluster Systems Handbook,* 1986.

[9] B. S. Dhillon, "The analysis of the reliability of multi-state device networks," PhD Thesis. Dept. of Industrial Engineering, University of Windsor, Windsor, Ontario, Canada, 1975.

[10] B. S. Dhillon, *Reliability Engineering in Systems Design and Operations,* Van Nostrand Reinhold Co., 1983.

[11] K. S. Trivedi, *Probability and Statistics with Reliability, Queueing and Computer Science Applications,* Prentice-Hall, 1982.

[12] O. C. Ibe, "Validation of the approximate availability analysis of VAXclusters," Internal Report, Digital Equipment Corporation, Andover, MA USA. 1988 February.

AUTHORS

Dr. Olive C. Ibe; Digital Equipment Corporation; 6 Tech Drive; Andover, Massachusetts 01810 USA.

Oliver C. Ibe (M'78) received the B Sc in electrical engineering from the University of Nigeria, Nsukka, Nigeria in 1975, the SM in electrical engineering and computer science from the Massachusetts Institute of Technology in 1979, the MBA from the Northeastern University, Boston, in 1980, and the ScD in electrical engineering from the Massachusetts Institute of Technology in 1981. From 1981 to 1983 he was a Post-doctoral Research Fellow at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York. From 1983 to 1987 he was an assistant professor of information and computer science at the Georgia Institute of Technology. He spent part of the summer of 1987 at the AT&T Bell Laboratories in Murray Hill, New Jersey. He is a Member of Technical Staff at the Digital Equipment Corporation, Andover, Massachusetts. His research interests include performance and reliability modeling of computer and communication systems, and queuing theory.

Dr. Ibe is a member of the ACM, Sigma Xi, and the Communication, Computer and Reliability societies of IEEE. He was the Assistant Editor of *Computer Networks* and *ISDN Systems* from 1985 to 1987, and is currently on the editorial board of that journal.

Richard C. Howe, Digital Equipment Corporation, 6 Tech Drive; Andover, Massachusetts 01810, USA.

Richard C. Howe received the BS in Electrical Engineering in 1971 and the MBA in 1981 from Northeastern University, Boston, Massachusetts. He joined Digital Equipment Corporation's Systems Reliability Engineering department in 1985 and currently manages the Advanced Systems Engineering group in reliability modeling and analysis of systems and networks.

Dr. Kishor S. Trivedi; Department of Computer Science; Duke University; Durham, North Carolina 27706 USA.

Kishor S. Trivedi (M'86, SM'87). For biography, see page of this issue.

Manuscript TR88-305 received 1988 March 7; revised 1988 June 27.

IEEE Log Number 25458 ◄ TR ►