CREDIBLE EXECUTION OF BOUNDED-TIME PARALLEL SYSTEMS WITH DELAYED DIAGNOSIS*

Raghu Shankar† and Daniel P. Miranker

Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712-1188

TR-91-30 September 1991

* This research was supported in part by the Office of Naval Research under contract N00014-86-K-0763 and by a grant from Texas Instruments.

† Department of Electrical and Computer Engineering.

Credible Execution of Bounded-Time Parallel Systems with Delayed Diagnosis

Raghu Shankar*
Daniel P. Miranker**
Department of Electrical and Computer Engineering*
Department of Computer Sciences**
University of Texas at Austin
Austin, Texas 78712

August 31, 1991

Abstract

This paper presents a forward recovery method for the fault-tolerant execution of parallel software systems on multicomputers such that faults are neither detected nor diagnosed until the fault prevents progress in the computation of the system. The method minimizes the communication and synchronization overhead required to verify the reliability of the system and consequently minimizes the impact of fault-tolerance on the throughput of the computation. We say the system is credible provided that the system is diagnosable and complete, where complete means that at least one copy of each process exists on a fault-free processor. We apply the method to the process structure deriving from parallel, bounded-time decision systems and show through an exact Markov analysis that the method will yield a very credible system. We then introduce a much simpler but approximate Markov model that facilitates credibility analysis over a larger range of parameters and applications. [1]

Keywords: fault-tolerance, multicomputer, parallel processing, real-time, expert-system, decision-system

[1]This research was supported in part by the Office of Naval Research under contract N00014-86-K-0763 and by a grant from Texas Instruments.

1 Introduction

The ease of constructing multicomputer systems due to advances in VLSI has motivated the interest in system-level fault-tolerance. The replicated processing elements and communication structure of a multicomputer intrinsically provide the hardware underpinnings of a fault-tolerant system. It remains to organize the replication and mapping of parallel processes to a multicomputer to support ongoing computation in the presence of hardware error.

Models for self-diagnosis of large systems has been inspired by the PMC Model [PMC67]. These models assume that there exists a set of tests that detect faults in a set of system elements and that no fault occurs during the application of the tests. On the basis of the responses to the stimuli, the outcome of the test is classified as *pass* or *fail*. A labeled graph, called a *syndrome*, then serves as the argument to a diagnosis algorithm. A system is called t-diagnosable if the system can self-diagnose all faulty units in the presence of up to t faulty units.

Malek introduced a comparison scheme for the diagnosis of multiprocessor systems such that syndromes based on a diverse set of task could be developed and analyzed[Mal80]. In these models each task is assigned to a pair of processors. A syndrome is generated on the basis of the set of completed tasks. An important assumption is that a faulty processor generates an incorrect result for everyone of the assigned tasks. Malek shows how to design systems with a minimum number of comparison edges and minimum number of comparisons to detect and locate a fault.

The major drawback of these techniques is that the detection process forces all processors to stop computation and synchronize and the recovery process forces a roll back and hence a recomputation. These features are not attractive when one considers large systems working in a real-time environment. Efforts must be made to break away from the need for regular synchronization and recomputation.

A second approach avoids complete system synchronization and diagnosis by replicating each task and verifying the correctness of an individual task with respect to itself. If the result of a task and its copy do not agree, additional copy(ies) of the task is executed to resolve the conflict [Agr85, LFA90]. These methods avoid global synchronization of the parallel system but require a minimum of three-fold redundancy in space and some redundancy in time to incorporate recovery measures.

We introduce the notion of *"statistical"* testing of processors and forward error recovery. As before processes will be replicated and distributed to distinct processors. But only the result of one task at a time will be compared for the detection of errors. The other processors carry on with their computation as usual. Only when the results of the duplicate copies of a task are inconsistent, are the results of the other tasks are used to diagnose the faulty processor. In contrast to *fast-fail* systems it is possible for a faulty processor to exist in the system until the processor's results are needed to corroborate the correctness of the

computation.

We show that the method applied to the natural mapping of bounded-time fault-tolerant decision systems to multicomputer systems will result in a reliable system where fault detection will be unobtrusive.

1.1 Definitions

A *faulty processor* means either the execution unit or/and the memory unit is/are faulty. A *signature* can be used to concisely represent the result of a process or checkpoint. Signatures can be compared between processors in finite time. The result of the comparison is a *link*. The link can be '0' or '1' based on whether the signatures are the same or different, respectively. Links represent logical connections. We assume the targeted multicomputer is logically fully connected using a fault-tolerant interconnection network and communication protocol [AS82, Pra82].

The logical graph formed as a result of comparing the signatures of the copies of a process is called a *subsyndrome*. A complete graph representing the comparison of all processes on all processors is called the *syndrome*. In this graph, a processor will be a node and a comparison will be a link. The set of processors that execute a common process are called *neighbors*.

Detectability of a faulty processor is the probability that the faulty processor will be detected in the current cycle. *Detectability of the system* is the rate at which faulty processors are detected. It is equal to the probability that at least one faulty processor will be detected in the current cycle. Given that a fault has been detected, *diagnosability* is the probability that the fault can be identified to a specific processor.

The system is is defined as *complete* when each process is running at least one fault-free processor. *Credibility* is defined as the probability that each process is running at least one fault-free processor and that processor can be identified. Hence, credibility is a combination of completeness and diagnosability. It is a restrictive definition of performability which is defined as the probability that the system performs at some level, L, at some instance of time, t.

1.2 BOCS Programs

Our work is motivated by a group project at University of Texas at Austin on the parallel, fault-tolerant execution of bounded-time decisions (BOCS) systems [BEG+90]. Our process model and technique derive directly from this application area. The method is applicable to the parallel execution of any collection of tasks such that the maximum time between task completions, or checkpoints, is bounded.

A BOCS program [Mok89] is a rule-based program similar to those used to develop both heuristic expert-system programs and formal declarative programs [BFKM85, CM88, Dij76]. A BOCS system is composed of three parts, a set of rules, a collection of state variables, and

an interpreter. The rules are of the form of *if-then* clauses where the *"if part"* is a predicate, or guard, on the values of the state variables and the *"then part"* contains a set of assignment statements that can update the values of the state variables. The interpreter repeats the following cycle until a fixed point is reached.

Match: For each rule, evaluate the predicate in its if-part using the current values of the state variables. Enumerate the set of satisfied rules to form the *conflict-set.*

Select: Choose a single rule from the conflict-set. For a BOCS program the rule is selected on the basis of a total priority order.

Act: Execute the actions in the then-part the chosen rule.

A set of analysis tools has been developed that constructively verify that a BOCS program will reach a fixed point in a bounded number of interpreter cycles [BEG+90]. [2] Work is ongoing to bound the execution time of the cycles [CKWC90]. We use the bound on the cycle time as the unit of time in our credibility analysis. The analysis tool also produces a schedule of rule firings. A schedule is a total priority order on the rules in the system such that forms the basis for selecting the firable rule each cycle.

2 The Fault Tolerant Execution of BOCS Programs

2.1 Normal Execution and Fault Detection

We will consider the parallel execution of BOCS programs derived from the *"full-distribution algorithm"* [Sto84]. The match-select-act cycle is replaced by a match-select-compare-act cycle in the fault-tolerant version. Each processor gets a copy of the rule-matcher and the state variables. Each rule is replicated k times, where $k \geq 2$. The rules and their copies are distributed among the processors with the condition that no more than one copy of a rule is allocated to a single processor.

Until a fixed point is reached,

1. Each processor, concurrently evaluates the predicates of the rules stored in its local memory and enumerates a subset of the conflict set.

2. Each processor then selects a winning rule from its subset of the conflict set.

3. A single winning rule is computed in log time by embedding a binary spanning tree on communications network such that each processor forms a leaf of the tree.

[2] BOCS programs are not Turing complete but are capable of capturing an important range of real-time processes monitoring applications.

(a) Each processor passes a triple up the tree. The triple contains the priority value of the winning rule, a unique rule identifier and a counter value initially set to 1.

(b) At each vertex of the tree the two values from the children are compared. If both children send the same rule identifier, one of the triples is updated with the sum of the two counters and the triple is sent up the tree. Otherwise, the triple containing the higher priority is sent up the tree.

(c) If a parent fails to receive a triple within a preset timeout period a null triple is assumed. A predictable and reasonable timeout period can be determined since the cycle time of the system is bounded.

4. If the counter value of the triple emitted by the top of the spanning tree equals k then we can assume that the winning rule has been computed correctly. If this is insufficient detail a more detailed signature can be added to the triple. By comparing such high-level results it is ensured that no common-mode failures exist.

5. If the counter value is less than k a fault has occurred and the machine enters a diagnostic and repair mode, described below.

6. A processor with a copy of the winning rule evaluates the actions of the then-part of the rule and the new values of the affected state variables are broadcast to the rest of the machine.

The method for fault detection checks only the accuracy of the multiple copies of the winning instantiation and ignores the remaining computation. Faults in the remaining processors will not be detected. In essence, a subsyndrome is developed each cycle by randomly sampling a set of processors. The computation continues, uninterrupted, until a fault prevents the accurate computation a winning rule. In this regard, the system is *credible* as long as there is at least one fault-free processor that computes the winning instantiation every cycle and that processor can be identified.

Note that during fault-free operation of a multicomputer the number of messages and topology of the communication is identical for the parallel-process structure derived from both the normal and fault-tolerant mapping of BOCS programs.

Although the algorithm above takes advantage of the implicit synchronization induced by the match-select-act cycle a similar comparison and diagnosis scheme can be created for any collection of parallel tasks, provided that the interarrival time between task checkpoints or task completions is bounded.

For example, in a distributed file systems, multiple copies of some of the files are maintained in order to protect these files against hardware failures. The signatures of key files can be compared pairwise at regular bounded intervals (especially after updates) to ensure high dependability of the system [FWA87]. In a real-time parallel object-oriented programming system, the execution time of any object will be bounded. Objects on their arguments can

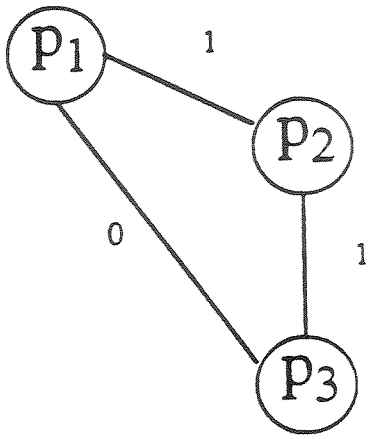

Figure 1: Example of one subsyndrome with $k = 3$; the winning instantiations are in processors p_1, p_2, and p_3.

be replicated and their results compared in isolation of the remaining object in the system [Agh86].

2.2 Diagnosis and Repair

Though we describe a statistical detection method, this work will consider only deterministic diagnosis algorithms [Mal80]. In the advent of a detected fault comparisons are performed between the k processors containing the selected rule and their logical neighbors. Refer to Fig. 1. It shows the subsyndrome formed when $k = 3$. The link between P1 and P3 is '0', hence both are fault-free. The link between P1 and P2 is '1'. This implies that at least one of them is faulty. Since, P1 is fault-free, P2 is faulty. This is also confirmed by the '1'-link between P2 and P3.

Each processor that computed the winning instantiation has a copy of the signatures from its neighbors. It independently diagnoses the syndrome and comes to its independent conclusion about the status of its neighbors. If the system is executing correctly then each fault-free processor will come to the same conclusion. This can be broadcast amongst the processors to ensure consistency of information.

One may not always be able to diagnose the initial subsyndrome. This happens when less than two fault-free processors are present in the initial subsyndrome. For example, in Fig. 1 if two out of three processors are faulty, there will be no '0' links. Hence, one cannot identify the fault-free and the faulty processors. (Note that two fault-free processors result in one '0'-link from which other processors can be diagnosed). Then one has to develop the entire syndrome. The comparison links formed by the matching of all the rule-copies are developed and the syndrome, so formed, is diagnosed.

After the system is diagnosed, the correct instantiation from a fault-free processor is fired,

the working memory is updated and communicated to the appropriate processors. The next cycle initiated.

3 Analysis

Required dependability can be achieved in two ways: (1) By changing the frequency of comparison of state or (2) By changing the number of copies of information. In the next two subsections, we shall present two Markov models to find the credibility of the system. The first model is a precise model but demands solving a set of linear equations that grow in proportion to the number of processors. The second model is approximated to a queueing model and gives a good estimation of credibility of the system with a large number of processors. The usual Markovian assumptions will be made.

To simplify the analysis and to be able to take advantage of earlier work on system diagnosis we consider a deterministic mapping of the rule-processes to processors instead of a random distribution. Distribute the original set of rules among the processors in round-robin fashion. Distribute a second copy of the rules in the same round-robin order but shifted by 1. Hence, the comparison edges will form a ring and the PMC model and Malek's comparison connection assignment can be extended for diagnosis [PMC67, Mal80]. We further assume that faulty processors are replaced from a collection of spares. Future work will consider the redistribution of work in faulty processors to other processors, graceful degradation of the system, along with probabilistic diagnosis methods [Mir88, FR89, DM84].

We use the bound on maximum cycle time of the application as the unit of time. Note that the duration of a cycle is not constant and can be much shorter than the bound. Hence, the credibility of the system will be better than the values we calculate. Typical values for the cycle time are expected to be 10 ms for an 8-processor systems and 1 ms for a 64-processor systems [MLar, MKB90].

The system consists of n processors and a BOCS program with r rules. Each processor has a Mean Time to Failure (MTTF), m, corresponding to which it has a constant failure rate, λ. The occurrence of faults is assumed to follow a Poisson Distribution. The fault may be a transient fault or a permanent fault. The lifetime of a transient fault is exponentially distributed and will be denoted by μ_y, where y comes from decay. The transient fault disrupts the working of the processor only during the time it exists and hence there may not be a computation of a correct local winning instantiation. After the fault decays, the processor reverts to its correct status. The processor then goes back to computing the local winning instantiation as before[BEG+90].

3.1 Exact Markov Analysis

The Markov diagram is shown in Fig. 3. F_i represents the state of the system with i undetected faulty processors. In each state, there is a probability, w_i, that the system is

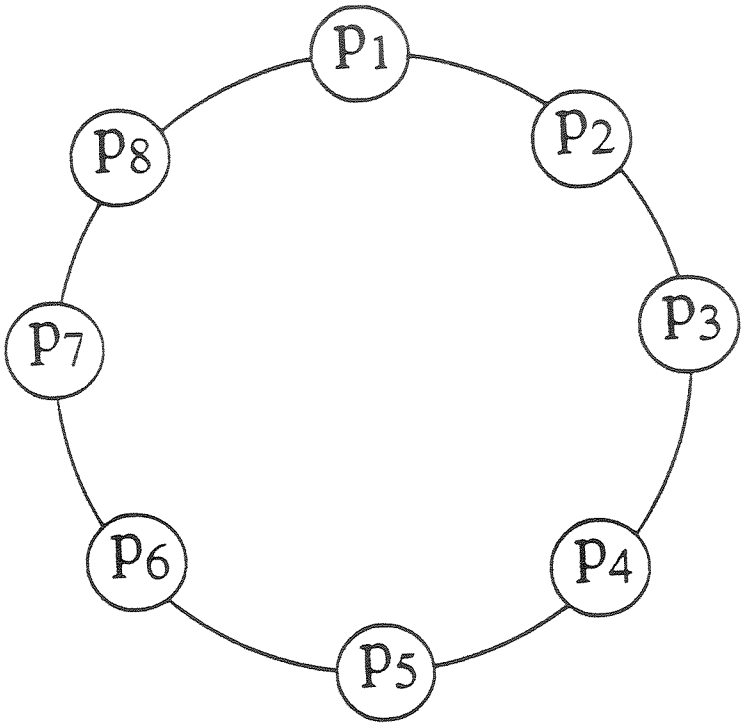

Figure 2: Mapping of rule-copies as a ring in an 8-processors system.

complete and, if so, the conditional probability, g_i, that the system is diagnosable. There is also a probability that faulty processors are detected in a cycle and that is given by μ_{di}. When a fault is detected all the faulty processors are recovered. When a fault decays the system goes to the state with one less faulty processor.

The system starts at F_0; every fault takes the system to another complete state or the *system failed* state, F_s. π_i is the probability of the system being in any one of the states, F_i. The system is *credible* as long as it is one of the complete states, F_i, where $0 \leq i \leq t$. In these states every rule is computed by at least one identifiable fault-free processor. Mathematically, it is given as:

$$C = \sum_{i=0}^{t} \pi_i$$

Credibility can be calculated by developing the Markov Equations as follows:

1. State F_0,

$$\lambda_0 \pi_0 = (\mu_{d1} + \mu_y)\pi_1 + \mu_{d2}\pi_2 + \cdots + \mu_{dt}\pi_t \tag{1}$$

2. State F_1,

$$(\mu_{d1} + \lambda_1)\pi_1 = \lambda_0 \pi_0 + 2\mu_y \pi_2 \tag{2}$$

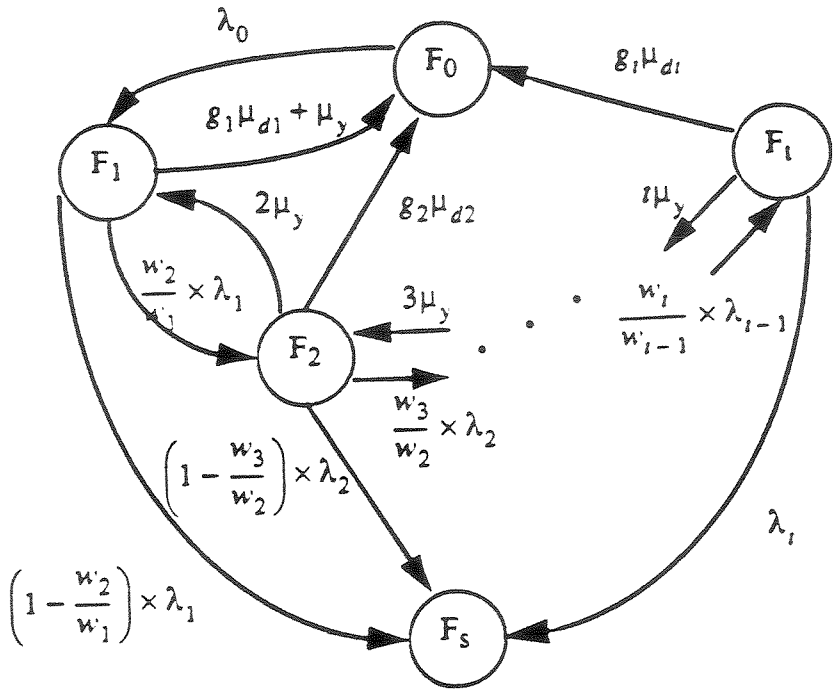

Figure 3: Transition Diagram for a system with n processors

3. State F_i, where $2 \leq i \leq t-1$,

$$(\mu_{di} + \tfrac{w_{i+1}}{w_i}\lambda_i + (1 - \tfrac{w_{i+1}}{w_i})\lambda_i + i\mu_y)\pi_i = \tfrac{w_i}{w_{i-1}}\lambda_{i-1}\pi_{i-1} + (i+1)\mu_y\pi_{i+1}$$

$$(\mu_{di} + \lambda_i + i\mu_y)\pi_i = \frac{w_i}{w_{i-1}}\lambda_{i-1}\pi_{i-1} + (i+1)\mu_y\pi_{i+1} \tag{3}$$

4. State F_t,

$$(\mu_{dt} + \lambda_t + t\mu_y)\pi_t = \frac{w_t}{w_{t-1}}\lambda_{t-1}\pi_{t-1} \tag{4}$$

5. State F_s,

$$\pi_s = (1 - \frac{w_2}{w_1})\lambda_0\pi_0 + (1 - \frac{w_3}{w_2})\lambda_1\pi_1 + \cdots + (1 - \frac{w_t}{w_{t-1}})\lambda_{t-1}\pi_{t-1} + \lambda_t\pi_t \tag{5}$$

Finally, individual state probabilities must meet the following criteria,

$$1 = \pi_0 + \pi_1 .... \pi_t + \pi_s \tag{6}$$

Solving the above equations we find the values of π_i, where $i = 1$ to t and s.

Lemma 1 *The Failure Rate,* $\lambda_i = \frac{n-i}{m}$.

We assume that the failure of processors is a Poisson process with a constant failure rate. The failure can be either due to a transient or a permanent fault. The Poisson Distribution is justified because the failure of components that form a processor is assumed to follow a Poisson process and the failure of a processor is an aggregate effect of the failure of a large number of its components. It has been observed that the Poisson processes appear very often where there is an aggregate effect.

The failure rate in any particular state is directly proportional to the number of fault-free processors in that state and inversely proportional to the MTTF of a processor. Thus,

$$\lambda_i = \frac{n-i}{m} \tag{7}$$

□

Lemma 2 *Completeness,* $w_i = \frac{(n-i-1)!(n-i)!}{(n-2i)!(n-1)!} \qquad i < \frac{n}{2}$.

Completeness is the probability that every rule is computed by at least one fault-free processor. The ring is complete as long as no two adjacent processors fail. It is given as

$$w_i = \frac{(n-i-1)!(n-i)!}{(n-2i)!(n-1)!} \qquad i < \frac{n}{2} \tag{8}$$

There are two cases:

1. $i > \frac{n}{2}$; in this case there will be at least one pair of adjacent processors that is faulty. Hence completeness is zero.

2. $i \leq \frac{n}{2}$; let F denote a faulty processor and N denote a non-faulty processor. Consider the ring as a horizontal string of processors. Pair every F with an N to get an FN pair. There will be i such pairs leaving $n-2i$ non-faulty processors. The number of ways this can be combined is $\begin{pmatrix} n-i \\ i \end{pmatrix}$. This does not include the case where the last processor can be an F. Then the first processor has to be an N. Again, form $i-1$ pairs of FN. There will be $n-2i$ N processors. The number of ways these can be combined are $\begin{pmatrix} n-i-1 \\ i-1 \end{pmatrix}$. The total number of cases in which the system is still complete is $\begin{pmatrix} n-i \\ i \end{pmatrix} + \begin{pmatrix} n-i-1 \\ i-1 \end{pmatrix}$. Let this equal a. Now, the total number of ways i processors can fail from n processors is $\begin{pmatrix} n \\ i \end{pmatrix}$. Let this equal b. Hence, completeness is $\frac{a}{b}$. Reducing the above expressions we get the result.

□

Repair Rate: Repair of a faulty processor can take place in two ways:

1. Decay Rate, μ_y, which will depend on the system hardware, environment and other factors.

2. Detectability, A fault is present in a processor which was to compute a winning instantiation and the system enters a diagnostic and repair mode.

Lemma 3 *Let there be i undetected faulty processors in a ring of n processors. Given that the system is complete, detectability for the system is given as*

$$\mu_{di} = 2\frac{i}{n}, \qquad i \leq t. \tag{9}$$

t is the diagnosability of the ring and is equal to $\lfloor \frac{n-1}{2} \rfloor$.

The proof is given as follows:
A faulty processor can be detected in two ways:

1. If it generates a winning instantiation in the cycle.

2. If it is detected as part of diagnosis of another faulty processor.

We shall neglect the second factor. This makes the calculation more conservative. The detectability of a failed processor is the sum of the activities of its rules. If we assume that the rules are distributed uniformly and the rule has an equal probability of firing then the activity of the failed processor is the product of its number of rule-copies and the activity of each rule. Each processor has $\frac{2r}{n}$ rule-copies and the activity of a rule is $\frac{1}{r}$. Hence,

$$d_1 = \frac{1}{r}.\frac{2r}{n}$$

$$d_1 = 2.\frac{1}{n}$$

Let one more processor fail; for the system to be complete the second faulty processor must not be adjacent to the first one. The detectability of this processor is same as calculated above, i.e., $2.\frac{1}{n}$. The detectability of the system is the sum of the activities of the two faulty processors. Hence,

$$d_2 = 2\frac{2}{n}.$$

In general, if i processors are faulty such that no two are adjacent, detectability of the system is the probability that at least one of the processors are detected in the current cycle and is the sum of the activities of the i processors. Hence,

$$d_i = 2\frac{i}{n}, \qquad i \leq t\ .$$

where $t = \lfloor \frac{n-1}{2} \rfloor$.

□

In the model, we shall equate the rate of detection in a given state, μ_{di}, to the detectability of the system in a cycle, i.e.,

$$\mu_{di} = d_i.$$

The rate of detection is directly proportional to the number of faulty processors, i, and indirectly proportional to the number of processors, n. Thus as the number of faults increases the rate of detection and hence the repair rate also increases. This makes the system inherently prone to self-repair. But as the number of processors increases the repair rate decreases. This is expected since the work done and hence the activity of each processor proportionately decreases.

Lemma 4 *Diagnosability for a ring, g_i,*
$g_i = 1$, *for* $i \leq t$,
$g_i = 0$, *for* $i > t$.
where, $t = \lfloor \frac{n-1}{2} \rfloor$ *See [PMC67]*

Since the system is credible in any of the complete states, F_i, we have

$$C = \sum_{i=0}^{t} \pi_i \tag{10}$$

$$or, \qquad C = 1 - \pi_s$$

Average Faults, f, is defined as the number of faults that is expected at one particular time. It is the sum of the products of the number of faults in state, F_i, and the probability, π_i, of being in the respective state. That is,

$$f = \textstyle\sum_{i=0}^{t} i \times \pi_i + \pi_s \sum_{i=t+1}^{n} i$$

Consider a 5-processor system with a 100 hour, constant failure rate per processor and a 3 second constant transient fault decay rate. The occurrence of permanent faults is included in the occurrence of transient faults. We assume that the 5-processor system executes at rate of 28 production system cycles per second. Translating the failure rate and the transient fault decay rate into a number of production system cycles, we have λ as $1/10^7$ per cycle per processor and a transient fault decay rate of 100 cycles, i.e., $\mu_y = 1/10^2$ cycles per transient fault. Substituting the above values and solving the equations we get credibility as $1 - 2.45 \times 10^{-12}$ and the average number of faults as 1.22×10^{-6}.

There is a testing overhead of 2 processors every cycle but we achieve a credibility of more than $1 - 10^{-12}$. Every cycle there is more than $1 - 10^{-12}$ probability that every rule is matched by at least one fault-free processor, resulting in a highly credible (dependable) system.

3.2 Approximate Markov model

It is seen that credibility is dependent on three parameters : the number of processors in the system, the frequency of comparisons, and the number of copies of the rule processes. Given a number of processors, credibility can be easily increased (decreased) by increasing (decreasing) the frequency of comparisons, increasing (decreasing) the number of copies. Given a failure rate, one can set the comparison rate and number of copies to get a desired value of credibility. In the previous model, one has to solve a set of simultaneous equations to evaluate, precisely, the credibility of the system. To understand system credibility as we increase the number of processors we introduce an approximate model to facilitate the analysis of large multicomputer systems. It will be seen that the results derived by approximation are close to those results derived from the precise analysis.

We shall make some more simplifying assumptions and equate the model to a queueing system. This will enable us to make use of the results developed for queueing systems [Kle75]. If the system spends most of its time in the low fault states the approximations will not change the results considerably.

We shall use the $M/M/\infty//n$: Finite Customer Population - *"Infinite"* Number of Servers with some modifications. The transition diagram is shown in Fig. 4. The corresponding diagram for the fault model is shown in Fig. 5 (a) and (b). (1) The number of processors in the system is equated to the capacity of the system, n. (2) The fault arrival rate, λ, is represented by the arrival of customers. The failure rate includes both transient and permanent faults. (3) A state with i faulty processors is a state with i customers. (3) The fault decay rate, μ_y, is represented by the departure of customers. The decay rate includes transient fault decay and fault detection followed by repair. (4) The average number of faults in the system is represented by the average number of customers.

An exponential arrival of faults and an exponential time for the repair or decay of faults is implicit in the model. An arbitrary number of repairs can proceed simultaneously and there is one server (or repairer) for each faulty processor.

The states from F_0 to F_t are complete with a certain probability. *Credibility* is the sum of the products of the probability in these states and the probability of completeness in the states.

$$C = \sum_{i=0}^{i=t} w_i \times p_i \tag{11}$$

Transitions from F_i to F_{i-1} are only possible if the state is complete, and hence, diagnosable. From lemma 2 states F_{t+1} to F_n $t = \lfloor \frac{n-1}{2} \rfloor$, represent states where there will be at least one pair of adjacent faulty processors, and the system will be incomplete and thus not credible. These states are combined together into one faulty state, F_s. See Fig. 5 (b).

The average number of faults is the average number of customers in the queueing system

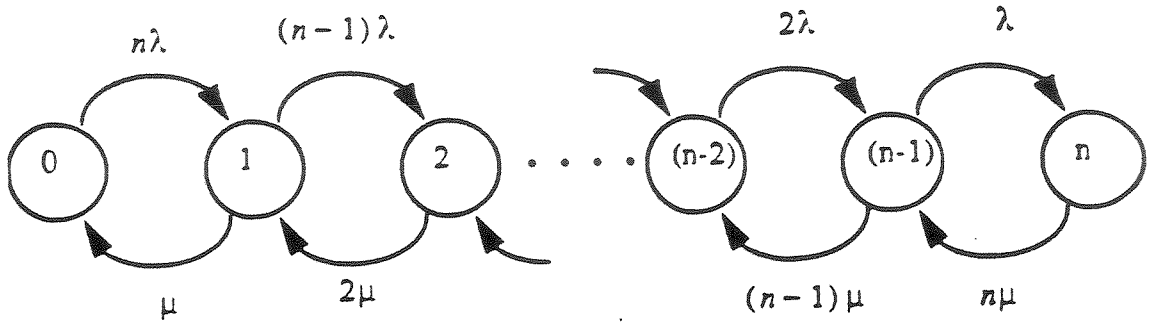

Figure 4: State-transition-rate diagram for *"infinite"*-server finite population system $M/M/\infty//n$.

[Kle75] :

$$f = \frac{n\frac{\lambda}{\mu}}{(1+\frac{\lambda}{\mu})} \tag{12}$$

The probabilities for p_0 and p_i is given as follows [Kle75]:

$$p_0 = \frac{1}{(1+\frac{\lambda}{\mu})^n} \tag{13}$$

$$p_i = \frac{(\frac{\lambda}{\mu})^i \begin{pmatrix} n \\ i \end{pmatrix}}{(1+\frac{\lambda}{\mu})^n} \tag{14}$$

Failure rate, probability of completeness, Diagnosability, and decay rate are as in the previous model. The detection of faults does not result in the repair of all the faults in the same cycle. Rather we have a repair such that the system progressively proceeds from high fault states to low fault states one step at a time. This repair rate due to detection is given by,

$$\mu_d = 2.\frac{1}{n} \tag{15}$$

Here the number 2 corresponds to the number of copies of each rule. It is clear that the detection rate increases with the number of copies of the rules but decreases with the number of processors in the system. The detection rate and the decay rate together gives the repair rate, μ.

$$\mu = \mu_d + \mu_y \tag{16}$$

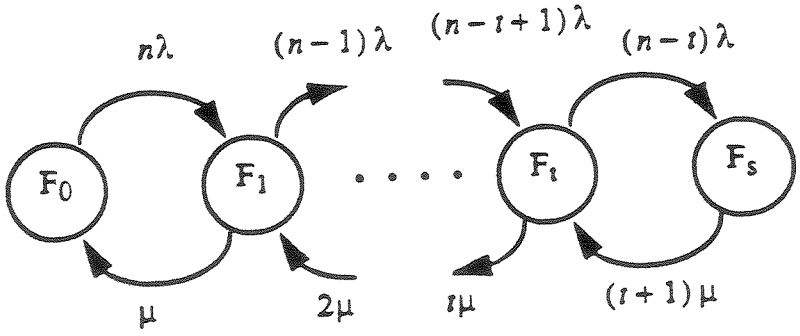

Figure 5: State transition-rate diagram for the fault model as a birth-death process

It is observed that,

1. Credibility is a function of p_i; to increase credibility one has to increase p_0, which is the state with no faults and decrease p_i ($i = 1$ to t, s). From Equation (8), for constant λ, the repair rate, μ has to be increased increase p_0. From Equation (11), we find that for constant μ_y, the detection rate, μ_d, has to be increased. By increasing the number of copies the detection rate, and hence, the credibility can be increased.

2. From Equation (8), it can be observed that the probability of the system being in higher fault states rapidly decreases.

3. From Equation (13) we see that f is linearly proportional to the number of processors in the system, n. f is also proportional to the ratio of the failure rate to the repair rate, $\frac{\lambda}{\mu}$. For constant λ, if the repair rate, μ doubles then the average number of faults reduces by a third.

We shall use the above results and a set of sample values to find the credibility of the system for different number of processors. As before, the failure rate per processor is assumed to be constant at approximately 100 hours and the transient fault decay rate is constant at approximately 3 seconds. Since the occurrence of transient faults is nearly 10 times the occurrence of permanent faults, it includes the occurrence of permanent faults.

Assuming that 4-processor and 5-processor systems execute at 28 cycles/sec the failure rate and the transient fault decay rate translate to failure rate, λ, as $1/10^7$ per production system cycle per processor and a transient fault decay rate of 100 cycles, i.e., $\mu_y = 1/10^2$ cycles per transient fault. The corresponding figures for 8-processor and 16-processor system is given in TABLE I.

TABLE I

n and cycles/sec, λ, and μ

States	Number of processors			
	4	5	8	16
cycles/sec	28	28	55	111
λ	10^7	10^7	2×10^7	4×10^7
μ	100	100	200	400

$\frac{\lambda}{\mu}$ is the ratio of the failure rate to the repair rate and is constant with increasing number of processors. This is because we assume linear speedup in the system with the number of processors. Thus the cycle times changes inversely proportional to the activity of the processors and the rate of testing per processor remains constant.

By using the above values, credibility for the above systems is given in TABLE II. Note that the result of the approximate model is reasonably close to the result derived from the

precise model.

TABLE II

n v/s $\frac{\lambda}{\mu}$, C, and f

Processors	$\frac{\lambda}{\mu}$	Credibility	Average Number of Faults
4	1.96×10^{-7}	$1 - 2.32 \times 10^{-13}$	7.84×10^{-7}
5	1.96×10^{-7}	$1 - 4.03 \times 10^{-13}$	1.22×10^{-6}
8	1.96×10^{-7}	$1 - 4.09 \times 10^{-13}$	1.57×10^{-6}
16	1.96×10^{-7}	$1 - 6.17 \times 10^{-13}$	3.14×10^{-6}

As the number of processors increase, credibility decreases marginally and average number of faults increases marginally. These are attributed to the following reasons:

1. The failure rate of the system increases in proportion to the number of fault-free processors.

2. The detection rate, and hence the repair rate, decreases with the number of processors in the system.

But, the decrease in credibility and increase in average number of faults is not linear with the number of processors. This decrease and increase is marginal and so we conclude that for large systems this *"statistical"* testing of processors leads to highly dependable systems.

4 Conclusions

For realistic mappings of BOCS programs to medium scale parallel processors the credibility of the system degrades only modestly as we consider additional processors and is greater than $1 - 10^{-12}$ for all mappings considered. The credibility of a system is a function of the ratio of the failure rate to the repair rate,$(\frac{\lambda}{\mu})$. The degradation is modest since processor load is inversely proportional to the number of processors. Thus, as the number of processors increases the rate of testing increases.

We have shown that simple duplication of the rule-processes in a BOCS program will be sufficient to create a highly credible bounded-time parallel decision system. When the system is fault free, the underlying error detection scheme requires no communication overhead beyond that required for a fault intolerant parallel implementation of the same program. We conclude that the statistical error detection method presented in this paper results in a highly credible system and by itself does not impinge on the throughput of the system. Interested readers may see [Mir88] for a method to ameliorate the overhead of the duplicate processes as well.

The repair rate, μ, is a function of detectability which is proportional to the number of copies of the processes. Credibility can then be expressed as a function of $\frac{\lambda}{k*\mu}$. For other application areas where the failure rate may be higher or the bound on the interarrival time of completed tasks is much longer than it is for BOCS programs a highly credible system may be obtained by increasing the number of replicated copies of an individual process.

The model presented in this paper assumed that hot-online spare processors were always available. We leave for future work the extension of the model to systems with non-repairable processors and incorporation of probabilistic diagnosis algorithms.[FR89, DM84]

References

[Agh86] Gul Agha. *Actors: A Model of Computation.* The MIT Press, Cambridge, Massachusetts, 1986.

[Agr85] P. Agrawal. RAFT: A Recursive Algorithm for Fault Tolerance. *IEEE*, 1985.

[AS82] G. B. Adams and H. J. Siegel. The Extra Stage Cube: A Fault-Tolerant Interconnection Network for Supercomputers. *IEEE Transactions on Computers*, pages 443–454, May 1982.

[BEG+90] J.C. Browne, A. Emerson, M. Gouda, D.P. Miranker, A. Mok, and L. Rosier. Bounded-time, Fault-tolerance, Rule-based Systems. In *Proceedings of the Goddard Conference on Space Applications of Artificial Intelligence*, 1990.

[BFKM85] L. Brownston, R. Farrell, E. Kant, and N. Martin. *Programming Expert Systems in OPS5.* Addison Wesley, Reading, Massachusetts, 1985.

[CKWC90] A. Mok C. K. Wang and A. Cheng. MRL: A Real-Time Rule-Based Production System. In *Proceedings of the 11th Real-Time Systems Symposium*, page to appear. IEEE, December 1990.

[CM88] K.M. Chandy and J. Misra. *Parallel Program Design: A Foundation.* Addison Wesley, Reading, Massachusetts, 1988.

[Dij76] E.W. Dijkstra. *A Discipline of Programming.* Prentice-Hall, 1976.

[DM84] A. T. Dabhura and G. M. Masson. An $O(n^{2.5})$ Fault Identification Algorithm for Diagnosable Systems. *IEEE Transactions on Computers*, pages 486–492, 1984.

[FR89] D. Fussel and S. Rangarajan. Probabilistic Diagnosis of Multiprocessor Systems. In *Fault-Tolerant Conference Symposium*, 1989.

[FWA87] W. K. Fuchs, K. L. Wu, and J. A. Abraham. Comparison and Diagnosis of Large Replicated Files. *IEEE Transactions on Software Engineering*, page 15, January 1987.

[Kle75] L. Kleinrock. *Queueing Systems, Volume I : Theory.* John Wiley, 1975.

[LFA90] J. Long, W.K. Fuchs, and J. A. Abraham. Forward Recovery Using Checkpointing in Parallel Systems. In *Proceedings of the IEEE International Conference on Parallel Processing*, volume I, pages 272–275. IEEE, 1990.

[Mal80] M. Malek. A Comparison Connection Assignment for Diagnosis of Multiprocessor Systems. pages 31–35, 1980.

[Mir88] Daniel P. Miranker. A High Level Language Approach to the Fault Tolerant Exectution of AI Expert Systems. Technical Report AI88-71, Artificial Intelligence Laboratory, University of Texas at Austin, March 1988.

[MKB90] D.P. Miranker, C.M. Kuo, and J.C. Browne. Parallelizing Compilation of Rule-Based Programs. In *Proceedings of the IEEE International Conference on Parallel Processing*, volume II, pages 247–251. IEEE, 1990.

[MLar] D.P. Miranker and B. J. Lofaso. The organization and performance of a treat based production system compiler. *IEEE Trans. on Knowledge and Data Engineering*, (to appear).

[Mok89] A. Mok. Formal Analysis of Real-Time Equational Rule-Based . In *Proceedings of the 10th Real-Time Systems Symposium*, pages 308–318. IEEE, December 1989.

[PMC67] F. P. Preperata, G. Metze, and R. T. Chien. On the Connection Assignment Problem of Diagnosable Systems. *IEEE Trans. on Elec. Computers*, EC-16:848–854, December 1967.

[Pra82] D. K. Pradhan. On a Class of Fault-Tolerant Multiprocessor Network Architecture. 1982.

[Sha90] R. Shankar. Credible Execution of Parallel Systems with Delayed Diagnosis. Master's thesis, Dept. of Electrical and Computer Engineering, University of Texas at Austin, 1990.

[Sto84] S. J. Stolfo. Five Parallel Algorithms for Production System Execution on the DADO Machine. In *Proceedings of the National Conference on Artificial Intelligence*, Austin, Texas, August 1984. AAAI.