The Average Availability of Multiprocessor Checkpointing Systems

James S. Plank Michael G. Thomason

November 3, 1998

Technical Report UT-CS-98-403
Department of Computer Science
University of Tennessee
November 3, 1998

See `http://www.cs.utk.edu/ plank/plank/papers/CS-98-403.html` for o ther information about this paper.

Abstract

Performance prediction of checkpointing systems in the presence of failures is a well-studied research area. The *average availability* is defined as a useful metric for uniprocessor checkpointing systems in a previous Technical Report [PT98]. This report introduces a discrete-parameter, finite-state Markov chain $\mathcal{M}$ to compute the availability for multiprocessor checkpointing systems. N is the number of processors in the system. Processors are interchangeable. At any time, each individual processor is either *nonfunctional* (failed and under repair) or *functional* (actively working on the task or standing-by as a spare). A specified minimum number a of the N processors must be functional in order for the system to work on a distributed task. The system does not use more than a processors but cannot compute with fewer than a. $\mathcal{M}$ is based on assumptions of independent exponential probability distributions for identically distributed interoccurrence times of failures and for identically distributed repair times. A separate continuous-parameter Markov chain $\mathcal{S}$ is used to compute some of the transition probabilities in $\mathcal{M}$. System availability is related to the speed-up obtained with multiple processors as a measure of real-time work on a long-running task. Finally, merging states to obtain a smaller Markov chain and some additional computations are briefly discussed.

1 Introduction and Assumptions

In reference [PT98], the authors describe a uniprocessor checkpointing system, and how to calculate the *availability* of such a system given certain assumptions. Their model of a checkpointing system is as follows:

- A uniprocessor is executing a long-running program. It may be in one of two states: functional, which means that it may devote all of its resources to executing the program, and failed, which means that it is inoperative.
- When a machine first becomes functional, it starts executing the program. Every I seconds, it initiates a *checkpoint* of the current state of the program. I is called the *checkpoint interval.*
- Once initiated, each checkpoint takes L seconds to complete. This is called the *checkpoint latency.* A checkpoint cannot be used for recovery until it has completed.
- If a failure occurs, the processor is unavailable to execute the program. Eventually, the processor is repaired, and may again run the program, but to do so, it must read the execution state stored in the most recently completed checkpoint. This is called *recovery.* Recovery from a checkpoint takes R seconds. This starts as soon as the processor becomes functional following a failure. R is called the *recovery time.* Once recovery is complete, the program resumes execution from the point in the program where the checkpoint was originally initiated. Also, once recovery completes, checkpointing is again resumed every I seconds.
- Each checkpoint takes C seconds of processing away from the program. This is called the *checkpoint overhead.* In real life, the overhead is distributed throughout the latency period, but for the purpose of modeling, it is applied to the beginning of the checkpoint. Thus, each checkpoint consists of C seconds of overhead, followed by $L - C$ seconds where the processor executes the application, but the checkpoint is not available for recovery following a failure. If the processor fails during that time period, it must recover from a previous checkpoint.
- The probability distribution functions of machine failure and repair are known. In [PT98], the repair times are assumed to be constant, but the failure distribution function may be anything.

Given the above parameters, the authors explain how to compute the long-term *availability* of the system. Availability is defined to be the fraction of time that the machine spends performing *useful work,* where useful work is time spent performing computation on the program that will never be redone due to a failure. In other words, this is the time spent executing the program before a checkpoint completes. If time is spent executing the program, but the machine fails before the next checkpoint completes, then that part of the program must be re-executed, and is therefore not useful. The goal of a checkpointing system is to maximize the availability in the presence of failures.

In this paper, we compute the availability of a parallel computing system. We assume that there N processors in the system. Processors are interchangeable. As above, at any point in time, each individual processor is either *nonfunctional* (failed and under repair) or *functional.*

The processors in the system cooperate to execute a parallel program. We assume that this program is designed so that it requires a fixed number, a, of processors to execute. The program does not use more than a processors and cannot continue with fewer than a. We assume that a may be set by the user at the beginning of program execution, and it does not change until the program is finished. This is typical of most parallel programs for distributed memory computing environments.

As long as there is no failure among the a processors (which we term *active*), these processors execute the program and checkpoint the global state of the computation at interval I. The act of checkpointing takes L seconds to complete, and induces an overhead of C seconds. In other words, in the absence of failure, each checkpoint adds C seconds to the running time of the program.

Processors that are functional but not executing the program are called *spares*. If an active processor fails and at least one spare is functional, a spare is chosen to replace the failed processor. Immediately, the computation on all active processors is halted, and they all (including the newly swapped-in spare) begin recovery from the most recently completed checkpoint. If a spare processor fails, or if a failed processor becomes functional while there are enough active processors, the active processors are not affected. If too many processors fail, the system cannot execute the program until there are once again a functional processors.

We assume that interoccurrence times of failures are independent and identically distributed (*iid*) as exponential random variables with the same failure rate $\lambda > 0$ for each functional processor. Likewise, times to repair are *iid* as exponential random variables with repair rate $\theta > 0$ for each nonfunctional processor. Occurrences of failures or repairs at exactly the same instant have probability 0 for the exponential probability laws.

In this paper, we introduce a *discrete-parameter, finite-state Markov chain* [Fel68, Par62] $\mathcal{M}$ to study the availability of the above parallel checkpointing system. Given the parameters N, a, C, L, R, I, λ and θ, we use $\mathcal{M}$ to determine the availability A of the parallel system. This is an asymptotic value that can be used to approximate the availability of executing a program with a long running time, or of many executions of a program with a shorter running time.

1.1 Utility and Reasonableness of the Assumptions

The determination of availability is useful in the following way. The user of a parallel checkpointing system is confronted with an important question: What values of a and I minimize the expected running time of my program? Using large values of a can lower the running time of the program due to more parallelism. However, it also exposes the program to a greater risk of not being able to run due to too few functional processors. Similarly, increasing I improves the performance of the program when there are no failures, since checkpointing overhead is minimized. However, it also exposes the program to a greater recomputing penalty following a failure. Thus, there may be an optimal combination of a and I to minimze the expected running time of a program in the presence of failures and repairs.

Suppose the user can estimate the failure-free running time RT_a of his or her program when employing a active processors and no checkpointing. Moreover, suppose the user can estimate C_a, L_a and R_a. Additionally, suppose that λ and θ are known. Then the user can select any value of a and I, and compute the availability $A_{a,I}$ of the system. Then the value $RT_a/A_{a,I}$ is an estimate of the program's average running time in the presence of failures. Thus, the user's question may be answered by choosing values of a and I that minimize $RT_a/A_{a,I}$. We give an example of such a calculation in Section 6.

Obviously, such a calculation is only useful if the underlying model has basis in reality. The model of the checkpointing system with parameters C, L, R and I mirrors several coordinated checkpointing systems that store their checkpoints to a centralized storage. There have been several implementations of such systems, for example the public-domain checkpointers MIST [CCG+95], CoCheck [Ste94, Ste96, PL96], Fail-Safe PVM [LFS93], as well as several unnamed checkpointers that have been used for research projects [EJZ92, EZ94, Pla96, PLP98].

The modeling of failures and repairs as *iid* exponential random variables has less grounding in reality. Although such random variables have been used in many research papers on the performance of uniprocessor and multiprocessor checkpointing systems (for example [You74, Gel79, Vai97, KS97, WF96]), the few studies that attempt to observe the nature of processor failures have shown that the time-to-failure and time-to-repair intervals are extremely unlikely to belong to an exponential distribution [LMG95, CS84, PE98].

Nonetheless, there are three reasons why performance evaluations based on exponential random values have utility. First, when failures are rare, independent events, their counts may be approximated by Poisson processes [BHJ92]. Poisson counts are equivalent to exponential interoccurrence times [BP75], meaning that that if failures are rare (with respect to I, C, R, L, etc), their TTF distribution may be approximated by an exponential. Since repairs are likely to be less rare, this effect is less significant. Second, if the true failure distribution has an increasing failure rate, rather than the constant failure rate of the exponential distribution, then the results of [PT98] and this paper provide a conservative (i.e. lower bound) approximation of the availability. For example, the LONG data set in reference [PE98] displayed an increasing failure rate. Third, simulation results on real failure data [PE98] have shown in the uniprocessor case that the determination of the optimal value of I using an exponential failure rate gives a good first-order approximation of the optimal value of I determined by the simulation.

Thus, in the absence of any other information besides a mean time to failure and a mean time to recovery for processors, the availability calculation in this paper is a reasonable indicator for selecting the values of a and I.

2 Overview of the Markov Model

In this paper, we introduce a *discrete-parameter, finite-state Markov chain* [Fel68, Par62] $\mathcal{M}$ to study the availability of the above parallel checkpointing system. States in $\mathcal{M}$ are defined by counts of active, spare, and nonfunctional processors. State-transition probabilities are based on likelihoods of failures and repairs. The *transition probability matrix* is the square matrix $\mathbf{P} = [p_{ij}]$ where p_{ij} is the conditional probability that the next state is j, given current state i. In addition to probability p_{ij}, transition arc $i \to j$ must also be labelled with the mean system uptime U_{ij} and the mean system downtime D_{ij} associated with that transition.

We use the long-run properties of $\mathcal{M}$ to compute A. $\mathcal{M}$ is a *recurrent chain* in which every state is reachable from every state in one or more transitions. Its well-defined, asymptotic properties can be found by standard methods [KS60, Par62]. In particular, the long-run, unconditional probability of occupancy of state i in terms of number of transitions is entry π_i in the unique solution of the matrix equation $\Pi = \Pi\mathbf{P}$ where $\sum_i \pi_i = 1, \pi_i > 0$. Once Π is determined, the long-term expected system uptime U and downtime D per arc may be determined by:

$$\begin{aligned} U &= \sum_{i,j} U_{ij}\pi_i p_{ij} \\ D &= \sum_{i,j} D_{ij}\pi_i p_{ij} \end{aligned}$$

And then the availability A may be determined by $U/(U+D)$.

Before specifying $\mathcal{M}$ in detail, we must describe a separate Markov chain $\mathcal{S}$ which models the status of inactive processors and provides values needed to fill in some of the transition probabilities in $\mathcal{M}$. To help keep these two chains separate, p's are used for probabilities in $\mathcal{M}$ and q's in $\mathcal{S}$.

3 Birth-Death Chain $\mathcal{S}$ for Inactive Processors

Let $s = N - a$ where N is the total number of processors and a is the number required to be active on a distributed task. Then s corresponds to the maximum number of *spare* processors that may be available to continue computation following the failure of an active processor. Suppose $s > 0$. During the time periods that the system has a processors active on the task, the remaining s processors may fail or be repaired independently. Both failure and repair of spares has no effect on the computation that is being performed by the active processors. It is only when an active processor fails that the number of functional spares becomes important.

Given a current number of functional spares i ($0 \le i \le s$), and a period of time t, we need to know the probability of there being exactly j functional spares at the end of time t, for $0 \le j \le s$. These probabilities may be calculated by a *continuous-parameter, finite-state, birth-death Markov chain* [CM72, Par62] $\mathcal{S}$.

$\mathcal{S}$ is distinct from the discrete-parameter Markov chain $\mathcal{M}$. Its $s+1$ states correspond to "m functional, f nonfunctional" for $m + f = s$ and are labelled in the form $[\frac{m}{f}]$. Its transition diagram is conventionally drawn with the parameters of the exponential probability laws on the arcs. See figure 1.

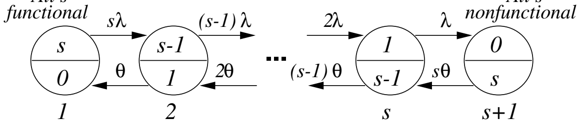

Figure 1: Birth-death Markov chain $\mathcal{S}$

The square matrix $\mathbf{R}$ of instantaneous transition rates is

$$\mathbf{R} = \begin{bmatrix} -s\lambda & s\lambda & 0 & 0 & \cdots & 0 & 0 & 0 \\ \theta & -((s-1)\lambda+\theta) & (s-1)\lambda & 0 & \cdots & 0 & 0 & 0 \\ 0 & 2\theta & -((s-2)\lambda+2\theta) & (s-2)\lambda & \cdots & 0 & 0 & 0 \\ & & & & \cdots & & & \\ 0 & 0 & 0 & 0 & \cdots & (s-1)\theta & -(\lambda+(s-1)\theta) & \lambda \\ 0 & 0 & 0 & 0 & \cdots & 0 & s\theta & -s\theta \end{bmatrix}.$$

By standard computation for this kind of process [CM72], the probability $q_{ij}(t)$ that $\mathcal{S}$ is in state j at time t, given starting in state i at $t=0$, is the $(ij)^{th}$ entry in the matrix

$$\begin{aligned} \mathbf{Q}(t) &= \exp(\mathbf{R}t) \\ &= \sum_{k=0}^{\infty} \mathbf{R}^k \frac{t^k}{k!} \end{aligned}$$

where the function $\exp(\mathbf{R}t)$ is the matrix exponential of $\mathbf{R}t$.

$\mathbf{Q}(\tau)$ is used to compute probabilities for status of the spare processors at the end of specific time periods τ. Three values of τ that will be important in the specification of M are:

τ_1: the unconditional mean time to failure ($MTTF$) among a active processors with *iid* exponential failures, defined as $\tau_1 = 1/(a\lambda)$.

τ_2: the length of time during which there must be no failure in order to leave the System Recovery Phase (described in Section 4, defined as $\tau_2 = R + I + L$.

τ_3: the conditional $MTTF$, given an unsuccessful attempt at recovery, defined as:

$$\begin{aligned} \tau_3 &= \frac{1}{a\lambda} - (R+I+L)\frac{e^{-a\lambda(R+I+L)}}{1-e^{-a\lambda(R+I+L)}} \\ &= \tau_1 - \tau_2 \frac{e^{-a\lambda\tau_2}}{1-e^{-a\lambda\tau_2}}, \end{aligned}$$

4 Completing Markov Chain $\mathcal{M}$

We can now return to the details of $\mathcal{M}$. First Figure 2 demonstrates an interval of the parallel program between two failures. The interval starts with a recovery, includes four successful checkpoints, and then ends with a failure before the fifth checkpoint begins. We define three *phases* that the system may be in:

- The *System Recovery Phase* encompasses the state of the system in the first $R + I + L$ seconds following a failure where there are at least a functional processors. The first R seconds are devoted to recovery from the most recent checkpoint, and the next I seconds are devoted to computation. In the final L seconds, a

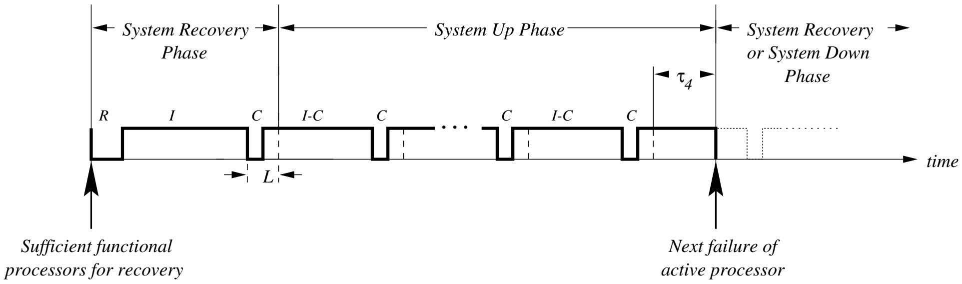

Figure 2: Illustrating System Phases

checkpoint is taken. It is only when that checkpoint completes that the system progresses past the System Recovery Phase to the System Up Phase. If a failure occurs before $R+I+L$ seconds have passed, then the system either starts a new System Recovery Phase (there still at least a functional processors), or moves to the System Down Phase (there are fewer than a functional processors).

If there is no failure in the System Recovery Phase, then I seconds of useful work are performed. If there is a failure in the System Recovery Phase, then zero seconds of useful work are performed, since any computation performed will be lost following the failure.

- The *System Down Phase* occurs whenever there are fewer than a functional processors. No useful work is performed in this phase.

- The *System Up Phase* is the state of the system from the time that it leaves the System Recovery Phase due to the completion of the first checkpoint, until the next failure. If M checkpoints complete while in the System Up Phase, then $M(I-C)$ seconds worth of useful work get performed in this phase.

4.1 Definition of the States of $\mathcal{M}$

Let $s > 0$ (the case $s = 0$ is a straightforward modification below); then the Markov Chain $\mathcal{M}$ has $N+s+1$ states organized into three groups, depicted in Figure 3.

- *System Up States* 1 through $s+1$ represent when the system is in the System Up Phase. They are labelled in the form $[\frac{U:a/m}{f}]$ to denote "a active, m functional spares, f nonfunctional spares" where $m+f=s$. These states are numbered 1 through $s+1$ with respective labels $[\frac{U:a/s}{0}]$ through $[\frac{U:a/0}{s}]$. The System Up states may only be entered when the system has been in a System Recovery state for $R+I+L$ seconds. The specific state entered depends on the number of functional spares at the time the state is entered. If m spares are functional, then state $[\frac{U:a/m}{f}]$ is entered. The state is exited when a failure occurs in an active processor. When that occurs, there may be any number of functional spares, from zero to s.

- *System Down States* $s+2$ through $N+1$ represent when the system is in the System Down Phase. This happens when there are fewer than a functional processors. These states are labelled $[\frac{D:m}{f}]$, for m functional,

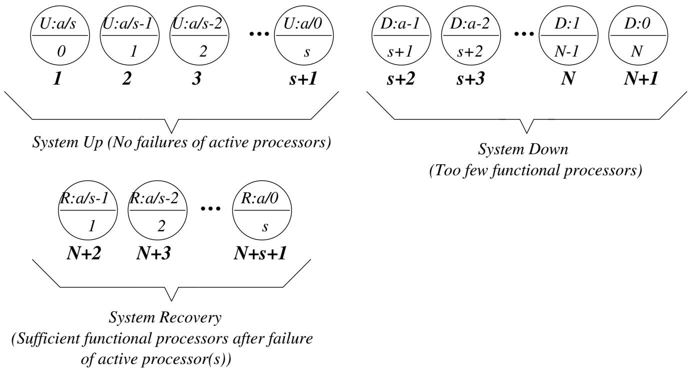

Figure 3: States of Markov chain $\mathcal{M}$.

f nonfunctional, where $m+f=N, m<a$. State $s+2$, labelled $[\frac{D:a-1}{s+1}]$ may be entered in three ways – from a System Up state after an active fails and there are no functional spares, from a System Recovery state after an active fails and there are no functional spares, and from state $[\frac{D:a-2}{s+2}]$ when a nonfunctional processor is repaired. State $[\frac{D:a-1}{s+1}]$ is exited either when a failed processor is repaired, or when a functional processor fails. State $[\frac{D:0}{N}]$ is only entered from state $[\frac{D:1}{N-1}]$ when the last active processor fails, and exits whenever any of its processors is repaired. For the other System Down states $[\frac{D:a-i}{s+i}]$, $1<i<a$, the states may be entered either from state $[\frac{D:a-i+1}{s+i-1}]$ when a functional processor fails, or from state $[\frac{D:a-i-1}{s+i+1}]$ when a failed processor is repaired. Like state $[\frac{D:a-1}{s+1}]$, these states are exited either when a failed processor is repaired, or when a functional processor fails.

- *System Recovery States* $N+2$ through $N+s+1$ represent when the system is in the System Recovery Phase. This happens when there are at least a functional processors, and less than $R+I+L$ seconds have passed since the most recent failure (or repair of the a-th active processor). The states are labelled $[\frac{R:a/m}{f}]$ to denote "a active, m functional spares, f nonfunctional spares." Thus, the labels are $[\frac{R:a/s-1}{1}]$ through $[\frac{R:a/0}{s}]$. Any System Recovery state may be entered from any System Up state or from any System Recovery state, whenever there is a failure. The specific state entered depends on the number of functional spares at the time of failure. Additionally, state $[\frac{R:a/0}{s}]$ may be entered from state $[\frac{D:a-1}{s+1}]$ when a failed processor becomes functional. System Recovery states are exited either when a failure occurs, or when $R+I+L$ seconds elapse with no failure. Note that when $s>0$, there is no state $[\frac{R:a/s}{0}]$. This is because all System Recovery states besides state $[\frac{R:a/0}{s}]$ must be entered due to a failure, and state $[\frac{R:a/s}{0}]$ would imply no failures.

Example. $\mathcal{M}$ for a 3-processor system with $a=2$ and $s=1$ has

two System Up states $\{1\equiv[\frac{U:2/1}{0}], 2\equiv[\frac{U:2/0}{1}]\}$,

two System Down states $\{3\equiv[\frac{D:1}{2}], 4\equiv[\frac{D:0}{3}]\}$, one System Recovery state $\{5 \equiv [\frac{R:2/0}{1}]\}$.

$\mathcal{M}$ for a 16-processor system with $a = 15$ and $s = 1$ has

two System Up states $\{1 \equiv [\frac{U:15/1}{0}], 2 \equiv [\frac{U:15/0}{1}]\}$,

fifteen System Down states $\{3 \equiv [\frac{D:14}{2}], 4 \equiv [\frac{D:13}{3}], \ldots, 17 \equiv [\frac{D:0}{16}]\}$, and

one System Recovery state $\{18 \equiv [\frac{R:15/0}{1}]\}$.

$\mathcal{M}$ for a 32-processor system with $a = 28$ and $s = 4$ has System Up states $\{1, \ldots, 5\}$, System Down states $\{6, \ldots, 33\}$, and System Recovery states $\{34, \ldots, 37\}$. ■

4.2 Definition of the Transition Probabilities and Weightings

All assumptions of *iid* exponential distributions apply. Given the values of failure rate $\lambda > 0$, repair rate $\theta > 0$, and checkpoint parameters *R, C, L,* and *I*, the transition probabilities, mean uptimes, and mean downtimes are computed for arcs exiting states as follows.

System Up states 1 through s+1. Consider a typical state $i \equiv [\frac{U:a/m}{f}]$ for $1 \le i \le s+1$. At the time of passage into state i, the system has a active processors, m spares, and f nonfunctional processors; but since repair or failure of inactive processors (that is, changes in m and f) do not immediately impact active processors working on the distributed task, the transitions out of state i correspond to active processor failure. The *MTTF* is $\tau_1 = 1/(a\lambda)$, at which time the conditional probabilities for the status of the inactive processors are the entries $q_{ij}(\tau_1)$ for row i in the matrix $\mathbf{Q}(\tau_1)$ for the birth-death chain $\mathcal{S}$, namely,

$q_{i1}(\tau_1)$ is the probability that all s inactive processors are functional,

$q_{i2}(\tau_1)$ is the probability that $s-1$ inactive processors are functional,

$\vdots$

$q_{is}(\tau_1)$ is the probability that one inactive processor is functional,

$q_{i,s+1}(\tau_1)$ is the probability that all s inactive processors are nonfunctional.

If an active processor fails and there is at least one functional spare, the transition is from state i to a System Recovery state[1]: for $1 \le k \le s$ and $j = N + 1 + k$

$$p_{ij} = q_{ik}(\tau_1).$$

If an active processor fails and there are no functional spares, the transition is to System Down state $s+2$:

$$p_{i,s+2} = q_{i,s+1}(\tau_1).$$

[1]Remember that subscripts on p refer to states in $\mathcal{M}$ and subscripts on q refer to states in $\mathcal{S}$.

Mean uptime U_{ij} and mean downtime D_{ij} are computed with reference to the fixed time interval I. Figure 2 illustrates a sequence or time segment beginning with a successful recovery and continuing through checkpoint intervals I in a System Up state until the next active processor failure.

The probability of the event "no active processor failure in an interval I" is $e^{-a\lambda I}$ and the probability of its complement is $1 - e^{-a\lambda I}$. Given current state i, those two events are the outcomes of a Bernoulli trial [Fel68] for which the mean number of trials until a failure is

$$M = \frac{e^{-a\lambda I}}{1 - e^{-a\lambda I}},$$

that is, M is the mean number of intervals I completed until failure causes an exit from state i. The mean uptime associated with transition $i \to j$ is $U_{ij} = M(I - C)$. The mean downtime $D_{ij} = MC + L + \tau_4$ includes MC for the successful checkpointing expected in a segment before failure, plus L and the conditional $MTTF$

$$\begin{aligned} \tau_4 &= \frac{1}{a\lambda} - I\frac{e^{-a\lambda I}}{1 - e^{-a\lambda I}} \\ &= \tau_1 - IM \end{aligned}$$

for the additional downtime expected due to failure.

System Down states s+2 through N+1. Look first at states $s+3$ through N, and consider a typical state $i \equiv [\frac{D:m}{f}]$. This state indicates that f processors are subject to repair rate θ and m are subject to failure rate λ, all as independent exponentials; thus, the cumulative distribution function is $F(t) = 1 - e^{-(m\lambda + f\theta)t}$. A property of this form of the exponential cdf is that, whenever an event does occur, the probability that it is a repair is $f\theta/(m\lambda + f\theta)$ and that it is a failure is $m\lambda/(m\lambda + f\theta)$ [CM72]. These two ratios are independent of the time an event occurs; hence, for a repair as first to occur

$$p_{i,i-1} = \frac{f\theta}{m\lambda + f\theta}$$

whereas for another failure before a repair

$$p_{i,i+1} = \frac{m\lambda}{m\lambda + f\theta}.$$

The uptime is 0 for these arcs. The mean downtime is the mean time to an event, $1/(m\lambda + f\theta)$.

State $s+2 \equiv [\frac{D:a-1}{s+1}]$ is similar except that its transition for a repair is to the System Recovery state $N+s+1 \equiv [\frac{R:a/0}{s}]$ because this repair is the event that a total of a processors has just become functional again:

$$p_{s+2,N+s+1} = \frac{(s+1)\theta}{(a-1)\lambda + (s+1)\theta},$$

$$p_{s+2,s+3} = \frac{(a-1)\lambda}{(a-1)\lambda + (s+1)\theta},$$

$$U_{s+2,s+3} = U_{s+2,N+s+1} = 0, \quad D_{s+2,s+3} = D_{s+2,N+s+1} = \frac{1}{(a-1)\lambda + (s+1)\theta}.$$

State $N+1 \equiv [\frac{D:0}{N}]$ is similar except that all processors have failed and there is no arc exiting to a higher numbered state, only an arc to state $N \equiv [\frac{D:1}{N-1}]$ with $p_{N+1,N} = 1$, $U_{N+1,N} = 0$, and $D_{N+1,N} = 1/(N\theta)$. Note that if $a = 1$, then state $N+1$ is the same as $s+2$ and is the only System Down state, in which case its exit-arc is to System Recovery state $N+s+1$ with probability 1.

System Recovery states N+2 through N+s+1. Transitions out of the System Recovery state $i \equiv [\frac{R:a/m}{f}]$ are based on the recovery time, $\tau_2 = R + I + L$. The probability of the event "no active processor failure during interval τ_2" (hence, a successful recovery) is $e^{-a\lambda\tau_2}$. The matrix $\mathbf{Q}(\tau_2)$ provides the probabilities for inactive processors when the recovery completes. Transitions are to the appropriate System Up states: for $1 \leq j \leq s+1$

$$p_{ij} = e^{-a\lambda\tau_2} q_{ij}(\tau_2).$$

The uptime for a successful recovery is I and the downtime is R.

The probability of failure that prevents a successful recovery is $1 - e^{-a\lambda\tau_2}$. The matrix $\mathbf{Q}(\tau_3)$ where τ_3 is the conditional $MTTF$

$$\tau_3 = \tau_1 - \tau_2 \frac{e^{-a\lambda\tau_2}}{1 - e^{-a\lambda\tau_2}}$$

gives the probabilities for the inactive processors at the expected time of an active processor failure. If all inactive processors are nonfunctional at the expected time of failure, the transition is to the System Down state $s+2$:

$$p_{i,s+2} = (1 - e^{-a\lambda\tau_2}) q_{i,s+1}(\tau_3).$$

Otherwise, the transition is to the appropriate System Recovery state to attempt the recovery again: for $1 \leq k \leq s$ and $j = N+1+k$

$$p_{ij} = (1 - e^{-a\lambda\tau_2}) q_{ik}(\tau_3).$$

The uptime for an unsuccessful attempt to recover is 0 and the mean downtime is the conditional $MTTF$ τ_3. This completes the specification of $\mathcal{M}$ for $s > 0$.

Example. The 5-state Markov chain $\mathcal{M}_1$ for $N = 3$, $a = 2$, and $s = 1$ has System Up states $\{1 \equiv [\frac{U:2/1}{0}], 2 \equiv [\frac{U:2/0}{1}]\}$, System Down states $\{3 \equiv [\frac{D:1}{2}], 4 \equiv [\frac{D:0}{3}]\}$, and System Recovery state $\{5 \equiv [\frac{R:2/0}{1}]\}$. Its transition probability matrix is

$$\mathbf{P} = \begin{bmatrix} 0 & 0 & q_{12}(\tau_1) & 0 & q_{11}(\tau_1) \\ 0 & 0 & q_{22}(\tau_1) & 0 & q_{21}(\tau_1) \\ 0 & 0 & 0 & \frac{\lambda}{\lambda+2\theta} & \frac{2\theta}{\lambda+2\theta} \\ 0 & 0 & 1 & 0 & 0 \\ e^{-2\lambda\tau_2} q_{21}(\tau_2) & e^{-2\lambda\tau_2} q_{22}(\tau_2) & (1 - e^{-2\lambda\tau_2}) q_{22}(\tau_3) & 0 & (1 - e^{-2\lambda\tau_2}) q_{21}(\tau_3) \end{bmatrix}.$$

Six arcs have nonzero mean uptimes:

$$U_{13} = U_{15} = U_{23} = U_{25} = M(I - C), \quad U_{51} = U_{52} = I.$$

The mean downtimes on the eleven arcs are

$$D_{13} = D_{15} = D_{23} = D_{25} = MC + L + \tau_4,$$

$$D_{34} = D_{35} = \frac{1}{\lambda + 2\theta},$$
$$D_{43} = \frac{1}{3\theta},$$
$$D_{51} = D_{52} = R, \quad D_{53} = D_{55} = \tau_3.$$

As a numerical illustration, suppose the $MTTF$ of a processor is 30 days and the $MTTR$ is 0.5 days, so $\lambda = 1/30$ and $\theta = 2$. Suppose $I = 2$ (checkpoint every two days), $R = L = 1/24$ (one hour), and $C = R/2$. Then $\tau_1 = 15, \tau_2 = 2.0833, \tau_3 = 1.0176, M = 7.0111, \tau_4 = 0.9778$, and written in matrix format:

$$\mathbf{P} = \begin{bmatrix} 0 & 0 & 0.0164 & 0 & 0.9836 \\ 0 & 0 & 0.0164 & 0 & 0.9836 \\ 0 & 0 & 0 & 0.0083 & 0.9917 \\ 0 & 0 & 1 & 0 & 0 \\ 0.8563 & 0.0141 & 0.0019 & 0 & 0.1278 \end{bmatrix},$$

$$\mathbf{U} = \begin{bmatrix} - & - & 13.8762 & - & 13.8762 \\ - & - & 13.8762 & - & 13.8762 \\ - & - & - & 0 & 0 \\ - & - & 0 & - & - \\ 2 & 2 & 0 & - & 0 \end{bmatrix},$$

$$\mathbf{D} = \begin{bmatrix} - & - & 1.1655 & - & 1.1655 \\ - & - & 1.1655 & - & 1.1655 \\ - & - & - & 0.2479 & 0.2479 \\ - & - & 0.1667 & - & - \\ 0.0417 & 0.0417 & 1.0176 & - & 1.0176 \end{bmatrix}.$$

Some of the computations with chain $\mathcal{S}$ are

$$\mathbf{R} = \begin{bmatrix} -0.0333 & 0.0333 \\ 2 & -2 \end{bmatrix},$$

$$\mathbf{Q}(\tau_1) = \mathbf{Q}(15) = \begin{bmatrix} 0.9836 & 0.0164 \\ 0.9836 & 0.0164 \end{bmatrix},$$

$$\mathbf{Q}(\tau_2) = \mathbf{Q}(2.0833) = \begin{bmatrix} 0.9838 & 0.0162 \\ 0.9694 & 0.0306 \end{bmatrix}. \blacksquare$$

4.3 When $s = 0$

The case of all processors active and no spares, that is, $s = 0$ and $N = a$, is simpler than $s > 0$ but gives a slightly different Markov chain $\mathcal{M}$. There is no separate chain $\mathcal{S}$. $\mathcal{M}$ has $N + 2$ states. The single System Up state is $1 \equiv [\frac{U:N/0}{0}]$, the System Down states are $2 \equiv [\frac{D:N-1}{1}]$ through $N + 1 \equiv [\frac{D:0}{N}]$, and the single System Recovery state $N + 2$ also has label $[\frac{R:N/0}{0}]$. Transition probabilities, mean uptimes, and mean downtimes follow similar specifications as above.

Example. The 5-state Markov chain $\mathcal{M}_2$ for $N = a = 3$ and $s = 0$ has

System Up state $\{1 \equiv [\frac{U:3/0}{0}]\}$,

System Down states $\{2 \equiv [\frac{D:2}{1}], 3 \equiv [\frac{D:1}{2}], 4 \equiv [\frac{D:0}{3}]\}$, and

System Recovery state $\{5 \equiv [\frac{R:3/0}{0}]\}$.

Its transition probability matrix is

$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{2\lambda}{2\lambda+\theta} & 0 & \frac{\theta}{2\lambda+\theta} \\ 0 & \frac{2\theta}{\lambda+2\theta} & 0 & \frac{\lambda}{\lambda+2\theta} & 0 \\ 0 & 0 & 1 & 0 & 0 \\ e^{-3\lambda\tau_2} & 1 - e^{-3\lambda\tau_2} & 0 & 0 & 0 \end{bmatrix}$$

The expected number of intervals of length I in state 1 is

$$M = \frac{e^{-3\lambda I}}{1 - e^{-3\lambda I}}.$$

Only two arcs have nonzero uptimes: $U_{12} = M(I - C)$ and $U_{51} = I$. The mean downtimes on the eight arcs include $D_{12} = MC + L + \tau_4, D_{51} = R$, and $D_{23} = 1/(2\lambda + \theta)$. As a numerical illustration using the same values of parameters as the previous example:

$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0.0323 & 0 & 0.9677 \\ 0 & 0.9917 & 0 & 0.0083 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0.8119 & 0.1881 & 0 & 0 & 0 \end{bmatrix},$$

$$\mathbf{U} = \begin{bmatrix} - & 8.9392 & - & - & - \\ - & - & 0 & - & 0 \\ - & 0 & - & 0 & - \\ - & - & 0 & - & - \\ 2 & 0 & - & - & - \end{bmatrix},$$

$$\mathbf{D} = \begin{bmatrix} - & 1.1025 & - & - & - \\ - & - & 0.4839 & - & 0.4839 \\ - & 0.2479 & - & 0.2479 & - \\ - & - & 0.1667 & - & - \\ 0.0417 & 1.0055 & - & - & - \end{bmatrix}. \blacksquare$$

5 Availability of Multiprocessor Checkpointing System

In reliability theory, an important metric for a system with units that fail and undergo repair is the probability that the system is operating at a specified time t [BP75]. This is the *availability at time t*, denoted $A(t)$. As a measure of average performance in the long-term,

$$A_{av} = \lim_{T \to \infty} \frac{1}{T} \int_0^T A(t)dt$$

which equals the limit $A = \lim_{t\to\infty} A(t)$ when both exist. For a multiprocessor checkpointing system, A is the fraction of time in the long-run that the system is actually at work on the distributed task. In other words, A is the total uptime during time span T, divided by T as $T \to \infty$.

We use the long-run properties of Markov chain $\mathcal{M}$ to compute A. $\mathcal{M}$ is a recurrent chain with well-defined, asymptotic properties [KS60, Par62]. In particular, the long-run, unconditional probability of occupancy of state i in terms of number of transitions is entry π_i in the unique solution of the matrix equation $\Pi = \Pi\mathbf{P}$ where $\sum_i \pi_i = 1, \pi_i > 0$.

In fact, the probabilities in vector Π are exactly the limiting relative frequencies of the states with respect to the number of transitions and are independent of the starting state of the Markov chain. If $\mathcal{M}$ is taken through n transitions selected according to the transition probabilities, and if n_i counts the occurrences of state i during those transitions, then

$$\pi_i = \lim_{n\to\infty} \frac{n_i}{n+1}.$$

Since each visit to state i is followed by probabilistic selection of an exit arc, the limiting relative frequency of occurrence of transition $i \to j$ is the long-run, joint probability $\pi_i p_{ij}$. Arc $i \to j$ has two associated random variables, one for the system uptime on the distributed task and another for the system downtime, with respective conditional mean values U_{ij} and D_{ij}. For a long-running task, $U_{ij}\pi_i p_{ij}$ is the expected contribution to uptime due to the relative frequency of arc $i \to j$ and $D_{ij}\pi_i p_{ij}$ is that arc's expected contribution to downtime. The availability A is the ratio of the *mean uptime per arc* to the *mean total time per arc*:

$$A = \frac{\sum_{i,j} U_{ij}\pi_i p_{ij}}{\sum_{i,j}(U_{ij} + D_{ij})\pi_i p_{ij}}.$$

Example. Consider the 5-state chain $\mathcal{M}_2$ for $N = a = 3$ and $s = 0$ with the same values for the parameters in the previous example, in particular, $I = 2$. We find

$$\Pi = [0.282, 0.3589, 0.0117, 0.0001, 0.3473],$$

$$\sum_{i,j} U_{ij}\pi_i p_{ij} = 3.0849, \quad \sum_{i,j} D_{ij}\pi_i p_{ij} = 0.5649$$

and

$$A = \frac{3.0849}{3.6498} = 0.8452.$$

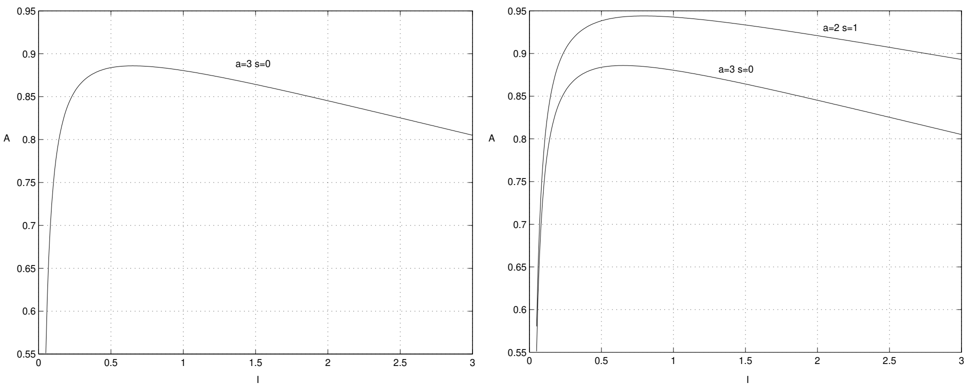

Figure 4: Plots of A vs I. Left: $\mathcal{M}_2$. Right: $\mathcal{M}_1$ and $\mathcal{M}_2$.

As shown by the plot of A vs I for $0.05 \leq I \leq 3$, the checkpoint interval $I = 2$ does not maximize A. At the optimal $I = 0.651$ we find

$$\Pi = [0.3101, 0.3449, 0.0112, 0.0001, 0.3337],$$

$$\sum_{i,j} U_{ij}\pi_i p_{ij} = 3.1069, \quad \sum_{i,j} D_{ij}\pi_i p_{ij} = 0.3999,$$

and the long-run fraction of real-time that all three processors are working on the task is

$$A = \frac{3.1069}{0.3999} = 0.886.$$

For comparison, suppose $N = 3, a = 2, s = 1$ with the same values of R, L, C, λ, and θ. The 5-state chain $\mathcal{M}_1$ for this case is also described in a previous example. The second plot has A vs I for both models $\mathcal{M}_1$ and $\mathcal{M}_2$. With two active processors and a spare, the optimal $I = 0.797$ gives

$$\Pi = [0.4747, 0.0066, 0.0082, 0.0001, 0.5104],$$

$$\sum_{i,j} U_{ij}\pi_i p_{ij} = 7.2293, \quad \sum_{i,j} D_{ij}\pi_i p_{ij} = 0.4287,$$

so the long-run fraction of time that two processors are working on the task (and one inactive processor is functional or nonfunctional) is

$$A = \frac{7.2293}{7.6579} = 0.944.$$

For completeness with the same values of R, L, C, λ, and θ, we remark that the case $N = 3, a = 1, s = 2$ has maximum availability $A = 0.9612$ for $I = 1.124$ where its 6-state Markov chain has

$$\Pi = [0.474, 0.0158, 0.0001, 0.0001, 0.4934, 0.0166]. \blacksquare$$

6 Availability vs. Speed-Up with Multiple Processors

As stated in the introduction, calculation of the availaibility of a multiprocessor checkpointing system is useful for determining the runtime parameters that may optimize the average execution time of a long-running program in the presence of failures. In this section, we present an example of such an optimization.

We assume a processing environment where $N = 8$ processors, and our program is one such as **PSTSWM** from [PLP98], which solves the nonlinear shallow water equations on a rotating sphere. We derive the following parameters from the **DISK-FORK** tests in reference [PLP98]. The checkpoint size is $(384.56 + 1.26a)$ MB, and the disk bandwidth is 0.1296 MB/sec. Therefore, the latency and recovery time are:

$$L_a, R_a = \frac{384.56 + 1.26a}{0.1296} \text{seconds}$$

Due to the **FORK** optimization, the checkpoint overhead is 0.0146 of the latency:

$$C_a = \frac{384.56 + 1.26a}{8.856} \text{seconds}$$

The running time of the program on one processor with no failures and no checkpointing is 270,769 seconds (3.13 days). When it is executed on a parallel environment, 85 percent of the computation is completely parallelizable, and the rest must run serially. Therefore, the running time of the program is:

$$RT_a = (270, 769)(.15 + .85/a)$$

Failure and recovery times are taken roughly from the **PRINCETON** data set in reference [PE98]: $\lambda = 1/30$ and $\theta = 2$.

We consider five values of a and s, and present their calculated optimal values of I and A in Table 1.

When $a = 8$, the program runs roughly 6 percent faster in the absence of failures than when $a = 7$. However, when failures are present, the program spends more time in the System Down state with $a = 8$ than $a = 7$ or 6. For optimal values of I, the expected running time of the program with checkpointing is roughly 8 percent faster when $a = 7$ than when $a = 8$.

7 Merging States in $\mathcal{M}$

In some cases, a Markov chain $\mathcal{M}'$ with fewer states than $\mathcal{M}$ yields the same calculation of availability. Consider the three groups of states—System Down, System Up, and System Recovery—as three disjoint sets.

a	s	L_a, R_a	C_a	Optimal I	$A_{a,I}$	RT_a	$RT_a/A_{a,I}$
8	0	3045.1 sec	44.6 sec	.062 days	.8457	69,385 sec	82,039 sec
7	1	3035.3	44.4	.067	.9684	73,494	75,893
6	2	3025.6	44.3	.072	.9721	78,974	81,238
5	3	3015.9	44.1	.078	.9757	86,646	88,806
4	4	3006.2	43.99	.088	.9793	98,154	100,230

Table 1: Calculating the optimal running time of a program in the presence of failures

The set of System Down states is entered by a transition to state $s+2$ and exited by the unique transition from state $s+2$ to System Recovery state $N+s+1$. Once passage into the set has occurred, the mean system downtime η accumulated until exit from the set is a function of a, s, λ, and θ but does not depend on checkpoint parameters R, L, C, and I. Therefore, for the purpose of computing availability, the entire set of System Down states can be merged together and represented by one state. That state has an exit-arc to the System Recovery state $N+s+1$ labelled with probability 1, uptime 0, and mean downtime η.

All pairs of System Up states i and j have $U_{ik} = U_{jk}$ and $D_{ik} = D_{jk}$ for all states k. If all pairs i and j also have $p_{ik} = p_{jk}$ for all states k, then there is no distinction among these states in terms of exit-arc probabilities, mean uptimes, or mean downtimes; consequently, for the purpose of evaluating availability, the System Up states can be merged together (or *lumped* [KS60]) into a single state. Likewise, all pairs i and j of System Recovery states have $U_{ik} = U_{jk}$ and $D_{ik} = D_{jk}$ for all states k; therefore, if all pairs i and j have $p_{ik} = p_{jk}$ for all states k, then the System Recovery states can be merged into a single state as well.

In the case of all these mergers, the resulting Markov chain $\mathcal{M}'$ in figure 7 has three states.

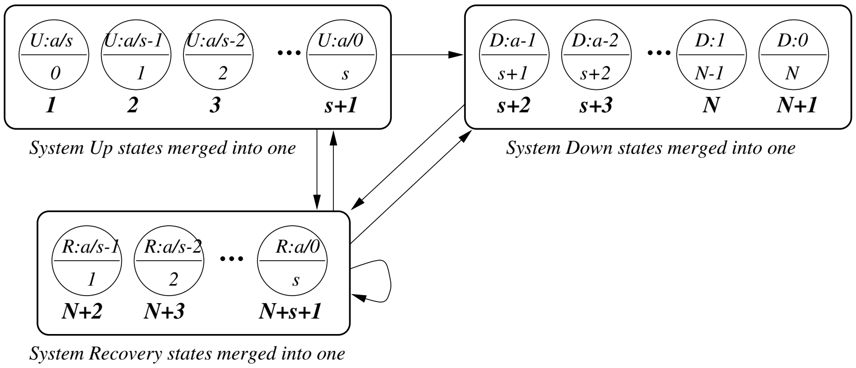

Figure 5: Representation of $\mathcal{M}'$ with merged states

The System Up states can be merged if and only if there are identical rows in the matrix $\mathbf{Q}(t)$ evaluated at $t = \tau_1$. For a Markov chain with the structure of $\mathcal{S}$, the rows in $\mathbf{Q}(t)$ do in fact converge to the same vector

$\mathbf{q} = [q_j]$, that is,

$$\lim_{t \to \infty} q_{ij}(t) = q_j$$

where q_j is independent of i and is the long-run fraction of time that $\mathcal{S}$ occupies state j [CM72]. $\mathbf{q}$ is called the *equilibrium distribution of* $\mathcal{S}$, and its values are directly computable as

$$q_j = \frac{\varrho_j}{\sum_{i=1}^{s+1} \varrho_i}$$

where $\varrho_1 = 1$ and

$$\varrho_j = \left(\frac{s-(j-2)}{j-1}\right)\left(\frac{\lambda}{\theta}\right)\varrho_{j-1}$$

for $j = 2, 3, \ldots, s+1$. If τ_1 is sufficiently large, then $\mathbf{Q}(\tau_1) \cong [\mathbf{q}]$ and the System Up states can be merged into one.

Example. For two-state $\mathcal{S}$ used with $\mathcal{M}_1$ in previous examples, $\mathbf{q} = [0.9836, 0.0164]$ and we have already computed

$$\mathbf{Q}(\tau_1) = \mathbf{Q}(15) = \begin{bmatrix} 0.9836 & 0.0164 \\ 0.9836 & 0.0164 \end{bmatrix}.$$

At the optimal $I = 0.797$, chain $\mathcal{M}_1$ has

$$\mathbf{P} = \begin{bmatrix} 0 & 0 & 0.0164 & 0 & 0.9836 \\ 0 & 0 & 0.0164 & 0 & 0.9836 \\ 0 & 0 & 0 & 0.0083 & 0.9917 \\ 0 & 0 & 1 & 0 & 0 \\ 0.9301 & 0.0129 & 0.0005 & 0 & 0.0565 \end{bmatrix}.$$

The three states in the smaller chain $\mathcal{M}_1'$ represent subsets of original states: its System Up state is $\{1, 2\}$, its System Down state is $\{3, 4\}$, and its System Recovery state is $\{5\}$. Computation with $\mathcal{M}_1'$ gives the same optimal $I = 0.797$ at which

$$\mathbf{P}' = \begin{bmatrix} 0 & 0.0164 & 0.9836 \\ 0 & 0 & 1 \\ 0.943 & 0.0005 & 0.0565 \end{bmatrix},$$

$$\mathbf{U}' = \begin{bmatrix} - & 14.2233 & 14.2233 \\ - & - & 0 \\ 0.797 & 0 & 0 \end{bmatrix},$$

and

$$\mathbf{D}' = \begin{bmatrix} - & 0.8184 & 0.8184 \\ - & - & 0.2514 \\ 0.0417 & 0.4359 & 0.4359 \end{bmatrix}.$$

The mean downtime for the arc $\{3,4\} \to \{5\}$ is computed with reference to the original chain $\mathcal{M}_1$ as

$$\begin{aligned}\eta &= \left(\frac{\lambda}{2\theta}\right)\left(\frac{1}{\lambda+2\theta}+\frac{1}{3\theta}\right)+\frac{1}{\lambda+2\theta}\\ &= 0.2514\end{aligned}$$

where $1/(3\theta)$ is the mean downtime on the one arc exiting state 4, $1/(\lambda+2\theta)$ is the mean downtime on both arcs exiting state 3, and the expected number of transitions $3 \to 4$ (each of them immediately followed by $4 \to 3$ with probability 1) before the first transition $3 \to 5$ is

$$\frac{\frac{\lambda}{\lambda+2\theta}}{\frac{2\theta}{\lambda+2\theta}} = \frac{\lambda}{2\theta}. \blacksquare$$

8 Other Calculations with $\mathcal{M}$

Several computations with a discrete-parameter, finite-state Markov chain give further information about the multiprocessor checkpointing system. Two additional computations with Π [KS60] are:

(1) The mean number of transitions until the next occurrence of state i, given current state i, is $1/\pi_i$.

(2) The mean number of times in state j between occurrences of state i is π_j/π_i.

Example. For $\mathcal{M}_1'$ in the previous example with optimal $I = 0.797$, the steady-state probability vector is

$$\Pi' = [0.4814, 0.0082, 0.5105]$$

and the mean number of transitions between occurrences of state $\{5\}$ is $1/0.5105 = 1.959$ which is the sum of entries in the vector

$$\frac{\Pi'}{0.5105} = [0.943, 0.016, 1].$$

Since transitions to the System Recovery state correspond one-to-one with swapping spares for failed processors, the mean number of transitions between these swaps is 1.959. Taking transition probabilities, uptimes, and downtimes into account, the expected value of the sum of system uptimes and downtimes between these swaps is computed to be 15.004. $\blacksquare$

Additional computations are possible.

9 Related Work

A few other researchers have studied performance prediction of multiprocessor checkpointing systems. Vaidya [Vai95] explores a *two-level* recovery scheme, where processors store checkpoints on both volatile storage that does not survive failures, and stable storage that does survive failures. In his system, two kinds of checkpoints are stored: *1-checkpoints*, that can survive the failure and repair of a single processor, and *n-checkpoints* which can survive the failure and repair of multiple processors. The conclusion that he draws is that a combination of 1-checkpoints and n-checkpoints results in better performance than relying on either 1-checkpoints or n-checkpoints exclusively. *Iid* exponentials are assumed. Vaidya's work differs from ours in that we only consider n-checkpoints, and Vaidya does not consider the issue of spare processors. It is a subject of future work to consider checkpointing protocols where R is a function of the number of active processor failures between the first active processor failure and successful recovery. Such protocols would then include a variety of diskless checkpointing protocols based on mirroring (as in [Vai95]), parity [PLP98] and Reed-Solomon coding [Pla97].

Wong and Franklin [WF96] study the performance of multiprocessor checkpointing systems in which the application either uses N processors, or it can reconfigure itself so that it makes use of however many processors are functional at the time. They assume that failures cannot occur during checkpointing or recovery. Our work differs in the relaxation of that assumption (which can be significant when programs exhibit a large checkpoint latency), in the use of spare processors for the large class of applications that cannot reconfigure, and in the separation of latency from overhead. It is a subject of future work to address the costs of reconfiguration upon failure or repair, given the assumptions of our system.

Finally, Kavanaugh and Sanders study of the performance of checkpoint consistency protocols based on limiting the skew of local clocks in multiprocessor systems [KS97]. Their analysis does not consider the issue of spares, repairs, or failures during checkpointing or recovery.

10 Conclusions

In this paper, we have have defined a way to compute the average availability of a multiprocessor checkpointing system, given values of checkpoint latency, recovey time, overhead, checkpoint interval, and *iid* exponential failure and repair rates. This availability calculation may be used to optimize two running time parameters of a parallel program: the number of active processors and the checkpoint interval. Future work in this area includes extending the checkpointing model to include more complex multiprocessor checkpointing algorithms such as diskless checkpointing [PLP98], and using failure and repair trace data to drive simulations of checkpointing systems given the same assumptions as in this paper.

11 Acknowledements

This material is based upon work supported by the National Science Foundation under grant CCR-9703390.

References

[BHJ92] A. D. Barbour, L. Holst, and S. Janson. *Poisson Approximation.* Clarendon Press (Oxford University), Oxford, UK, 1992.

[BP75] R. E. Barlow and F. Proschan. *Statistical Theory of Reliability and Life Testing.* Holt, Reinhart, and Winston, Inc., NY, 1975. Republished by TO BEGIN WITH, Silver Spring, MD, 1981.

[CCG+95] J. Casas, D. L. Clark, P. S. Galbiati, R. Konuru, S. W. Otto, R. M. Prouty, and J. Walpole. MIST: PVM with transparent migration and checkpointing. In *3rd Annual PVM Users' Group Meeting*, Pittsburgh, PA, May 1995.

[CM72] D. R. Cox and H. D. Miller. *The Theory of Stochastic Processes.* Chapman and Hall Ltd., London, UK, 1972.

[CS84] L. H. Crow and N. D. Singpurwalla. An empirically developed fourier series model for describing software failures. *IEEE Transactions on Reliability*, R-33:176–183, June 1984.

[EJZ92] E. N. Elnozahy, D. B. Johnson, and W. Zwaenepoel. The performance of consistent checkpointing. In *11th Symposium on Reliable Distributed Systems*, pages 39–47, October 1992.

[EZ94] E. N. Elnozahy and W. Zwaenepoel. On the use and implementation of message logging. In *24th International Symposium on Fault-Tolerant Computing*, pages 298–307, Austin, TX, June 1994.

[Fel68] W. Feller. *An Introduction to Probability Theory and Its Applications (Third Edition).* John Wiley & Sons, Inc., NY, 1968.

[Gel79] E. Gelenbe. On the optimum checkpoint interval. *Journal of the ACM*, 26:259–270, April 1979.

[KS60] J. G. Kemeny and J. L. Snell. *Finite Markov Chains.* Van Nostrand, Princeton, NJ, 1960. Republished by Springer-Verlag, NY, 1976.

[KS97] G. P. Kavanaugh and W. H. Sanders. Performance analysis of two time-based coordinated checkpointing protocols. In *1997 Pacific Rim International Symposium on Fault-Tolerant Systems (PRFTS'97)*, Taipei, Taiwan, December 1997.

[LFS93] J. León, A. L. Fisher, and P. Steenkiste. Fail-safe PVM: A portable package for distributed programming with transparent recovery. Technical Report CMU-CS-93-124, Carnegie Mellon University, February 1993.

[LMG95] D. Long, A. Muir, and R. Golding. A longitudinal survey of internet host reliability. In *14th Symposium on Reliable Distributed Systems*, pages 2–9, Bad Neuenahr, September 1995. IEEE.

[Par62] E. Parzen. *Stochastic Processes*. Holden-Day, San Francisco, CA, 1962.

[PE98] J. S. Plank and W. R. Elwasif. Experimental assessment of workstation failures and their impact on checkpointing systems. In *28th International Symposium on Fault-Tolerant Computing*, pages 48–57, Munich, June 1998.

[PL96] J. Pruyne and M. Livny. Managing checkpoints for parallel programs. In *Workshop on Job Scheduling Strategies for Parallel Processing (IPPS '96)*, 1996.

[Pla96] J. S. Plank. Improving the performance of coordinated checkpointers on networks of workstations using RAID techniques. In *15th Symposium on Reliable Distributed Systems*, pages 76–85, October 1996.

[Pla97] J. S. Plank. A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems. *Software – Practice & Experience*, 27(9):995–1012, September 1997.

[PLP98] J. S. Plank, K. Li, and M. A. Puening. Diskless checkpointing. *IEEE Transactions on Parallel and Distributed Systems*, 9(10):972–986, October 1998.

[PT98] J. S. Plank and M. G. Thomason. The average availability of uniprocessor checkpointing systems, revisited. Technical Report CS-98-400, University of Tennessee, August, 1998.

[Ste94] G. Stellner. Consistent checkpoints of PVM applications. In *First European PVM User Group Meeting*, Rome, 1994.

[Ste96] G. Stellner. CoCheck: Checkpointing and process migration for MPI. In *10th International Parallel Processing Symposium*, April 1996.

[Vai95] N. H. Vaidya. A case for two-level distributed recovery schemes. In *ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, Ottawa, May 1995.

[Vai97] N. H. Vaidya. Impact of checkpoint latency on overhead ratio of a checkpointing scheme. *IEEE Transactions on Computers*, 46(8):942–947, August 1997.

[WF96] K. F. Wong and M. Franklin. Checkpointing in distributed systems. *Journal of Parallel & Distributed Systems*, 35(1):67–75, May 1996.

[You74] J. S. Young. A first order approximation to the optimum checkpoint interval. *Communications of the ACM*, 17(9):530–531, September 1974.