Optimal Fault-Tolerant Computing on Two Parallel Processors

John Bruno
Computer Science Dept.
University of California
Santa Barbara, California 93106

E. G. Coffman, Jr.
AT&T Bell Laboratories
Murray Hill, New Jersey 07974

October 10, 1994

ABSTRACT

Suppose two identical processors, both subject to random failures, are available for running a single job of given duration τ. The failure law, whose mean is normalized to 1 for convenience, is operative only while a processor is active. To guard against the loss of accrued work due to a failure, checkpoints can be made, each requiring time δ; a successful checkpoint saves the state of the computation, but failures can also occur during checkpoints. The problem is to determine how best to schedule checkpoints if the goal is to maximize the probability that the job finishes before both processors fail.

We solve this problem under the assumption of an exponential failure law. In particular, for given τ and δ we show how to determine an integer $k \geq 0$ and time intervals $I_1, \ldots, I_{k+1}$ such that an optimal procedure is to run the job on one machine, checkpointing at the end of each interval $I_j, j = 1, \ldots, k$, until either the job is done or a failure occurs. In the latter case, the remaining processor resumes the job starting in the state saved by the last successful checkpoint; the job then runs until it completes or until the second processor also fails. We give an explicit formula for the maximum achievable probability of completing the job for any fixed $k \geq 0$. An explicit result for k_{opt}, the optimum value of k, seems out of reach; however, we give upper and lower bounds on k_{opt} that are remarkably tight; they show that only a few values of k need to be tested in order to find k_{opt}. We also derive the asymptotic estimate

$$k_{opt} - \sqrt{2\tau/\delta} = O(1) \quad \text{as} \quad \delta \to 0 \ .$$

Finally, we calculate conditional expected job completion times and discuss several open problems.

Optimal Fault-Tolerant Computing on Two Parallel Processors

John Bruno
Computer Science Dept.
University of California
Santa Barbara, California 93106

E. G. Coffman, Jr.
AT&T Bell Laboratories
Murray Hill, New Jersey 07974

1. Introduction

Assume we are given a single job with a known processing time $\tau > 0$, and that we are to run the job on a system of two identical processors subject to failures. A processor can fail only while it is running a job, not when it is idle. Times to failure of the processors are independent, identically distributed random variables with an exponential failure law $F(t)$. For convenience, we take the mean failure time as the time unit so that $F(t) = 1 - e^{-t}$, $t \geq 0$.

The objective is to schedule the job so as to maximize its *completion probability*, *i.e.*, the probability that the job completes before both processors fail. To reduce the amount of lost work owing to failures, and to increase the completion probability, *checkpoints* may be introduced. A checkpoint on a processor running the job simply saves the state of the computation and makes it available to the other processor. The checkpoint procedure, also subject to a processor failure, requires a fixed amount of time denoted by $\delta > 0$.

To illustrate the use of a single checkpoint, identify one of the processors as primary and the other as back-up. For some given x, $0 \leq x \leq \tau$, we start the job on the primary processor, attempt to run it for x time units, and then attempt to checkpoint the computation during $[x, x+\delta]$. If a failure occurs in $[0, x+\delta]$, the job is simply restarted on the back-up processor; otherwise, the job is continued on the primary processor in an attempt to complete the remaining $\tau - x > 0$ time units by time $\tau + \delta$. If this latter attempt fails, then the back-up processor repeats the attempt, *i.e.*, it starts in the checkpointed state and attempts to complete the last $\tau - x$ time units.

The above policy can be extended in the obvious way to any number of checkpoints on the primary processor. Checkpoints are made at appropriate intervals to guard against the loss of too much accrued work in the event of failure. The back-up processor is used only if the primary processor fails, in which case it resumes the job at the most recent successful checkpoint. We will see that, for any $\delta > 0$, the use of checkpoints entails a compromise, since the probability that the primary processor fails before completing the job increases with the number of checkpoints. Thus, our specific goal will be to find the number of checkpoints and the times they are made which maximize the completion probability.

It is easy to see that the completion probability can not be increased by adopting a policy not of the above sequential type. In particular, running the processors in parallel can not increase the completion probability. Indeed, the completion probability will decrease if the back-up processor is started before the last checkpoint, if any, is successfully made on the primary processor. On the other hand, finishing a schedule by running both processors after a last successful checkpoint has no effect on the completion probability and has the advantage of reducing the conditional job completion time, given that the job completes. The conditional job completion time under this variant is studied in Section 4; until then we keep with the simpler policies that use the processors in sequence, the back-up processor being used only in the event of a failure on the primary processor.

To illustrate calculations, let $Q_k(\tau) = Q_k(\tau, \delta)$ denote the completion probability under an optimal k-checkpoint schedule. Trivially

$$Q_0(\tau) = 1 - (1 - e^{-\tau})^2 = 2e^{-\tau} - e^{-2\tau}, \quad \tau \geq 0 \; . \tag{1.1}$$

For one checkpoint (see Fig. 1), we have

$$Q_1(\tau) = \sup_{0 \leq x \leq \tau} [(1 - e^{-(x+\delta)})e^{-\tau} + e^{-(x+\delta)} Q_0(\tau - x)], \quad \tau \geq 0 \; , \tag{1.2}$$

where the bracketed function is the completion probability assuming the checkpoint is made at time x. To see this, note that $(1 - e^{-(x+\delta)})e^{-\tau}$ is the joint probability that

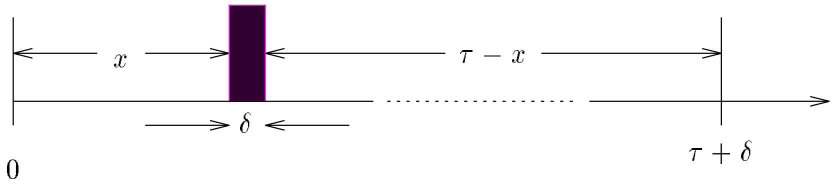


Figure 1: One Checkpoint

(i) the primary processor fails to complete x units of service plus the checkpoint, and

(ii) the back-up processor completes the job starting from scratch. The second term, $e^{-(x+\delta)}Q_0(\tau - x)$, is the joint probability that (i) the primary processor completes the checkpoint, and (ii) the processors, without making any additional checkpoints, complete the remaining $\tau - x$ time units required by the job.

Routine calculus shows that the supremum in (1.2) is reached at the unique point $x = \tau/2$, so that

$$Q_1(\tau) = (1 - e^{-(\tau/2+\delta)})e^{-\tau} + e^{-(\tau/2+\delta)}Q_0(\tau/2) \ . \tag{1.3}$$

A comparison of (1.1) and (1.3) shows that one checkpoint is better than none, *i.e.*, $Q_1(\tau) > Q_0(\tau)$, if and only if

$$\delta < \ln \frac{2}{1 + e^{-\tau/2}} \ . \tag{1.4}$$

For δ sufficiently small, more than one checkpoint will be better than either one checkpoint or no checkpoints. Indeed, in Section 3 we show that, for fixed τ, the optimal number of checkpoints grows like $\sqrt{2\tau/\delta}$ as $\delta \rightarrow 0$. The limit itself is artificial if one considers a primary processor checkpointing infinitely often on a set of measure 0 and dense in $[0, \tau]$. But in this set-up the probability of failures during checkpoints is 0, and the work that has to be repeated on the back-up processor in the event of a failure has measure 0. Thus, the completion probability is

$$1 - \int_{x=0}^{\tau} e^{-x}(1 - e^{-(\tau - x)})dx = (\tau + 1)e^{-\tau} \ . \tag{1.5}$$

Section 3 verifies analytically that the optimum completion probability tends to (1.5) as $\delta \rightarrow 0$.

Much of the literature in fault-tolerant scheduling deals with a repairable processor so that the probability of completing a job is 1. Checkpoints are still used to avoid losing too much work, but the objective is to minimize the expected completion time of the job [2, 5, 6, 7]. Recently, fault tolerant scheduling in a multiprocessor, online environment has been studied in [1]. In this model there are m processors and a collection of n jobs whose processing times are known only when they complete their processing. The processors are subject to either permanent or transient failures and various objective functions are considered. Competitive analysis [8] is used to evaluate possible algorithms. In the case of permanent failures it is assumed that no more than a constant fraction of the processors fail and so all the jobs are eventually completed.

The paper by Geist, *et al.* [10] considers job completion probability in a single-processor system in which the processor can undergo a limited number N of repairs, where N is a random variable. Checkpoints are spaced at uniform intervals τ/k where k is the number of checkpoints. Various alternatives for N, the number of repairs, are considered. For example, if timeliness is an issue, then the number of repairs could depend on the total time spent in processing the task and repairing the processor. A similar model is considered in [4] in which a more general checkpointing strategy is allowed and each failure has probability $1 - a$ of being permanent. The objective is to maximize the probability of completing the job before the first permanent failure of the processor. The checkpointing strategies depend on the distribution of the time-to-failure random variable; however, in many cases, including the exponential failure law, optimal checkpointing is done at intervals which are, for the most part, uniformly spaced.

Section 2 fixes k and treats the general problem of determining the optimal schedule of k checkpoints, *i.e.*, the durations of the $k+1$ time intervals bounded by checkpoints, with the first and the last such interval beginning and ending at times 0 and $\tau + k\delta$, respectively. We give explicit formulas for the durations of these intervals, to be called *checkpoint intervals*, in the optimal k-checkpoint schedule.

Explicit results for the optimum number of checkpoints as a function of δ and τ appear to be out of reach. For this reason, Section 3 turns to bounds. Upper and lower bounds are derived which are remarkably tight for most parameter values of practical interest. This means that the numerical search for the optimum number of checkpoints usually tests very few possible values.

Section 4 calculates the conditional expected job completion time, given that the job completes, and studies the trade-off between this metric and the completion probability. Numerical results indicate that in many circumstances a small sacrifice (decrease) in the completion probability yields a substantial decrease in the conditional expected completion time.

2. How to Use k Checkpoints

In this section we determine an optimal k-checkpoint schedule for any fixed k. Because of the memoryless property of the exponential distribution, we can express $Q_k(\tau)$ as the solution to a finite-stage stochastic dynamic program [3]; if the primary processor successfully completes the first checkpoint in an optimal k-checkpoint schedule, then the schedule after the first checkpoint is an optimal $(k-1)$-checkpoint schedule on the remaining time. For $k \geq 1$ we have,

$$(2.1) \qquad Q_k(\tau) = \sup_{0 \leq x \leq \tau} [(1 - e^{-(x+\delta)})e^{-\tau} + e^{-(x+\delta)} Q_{k-1}(\tau - x)], \quad \tau \geq 0 \ .$$

The bracketed term in (2.1) is the sum of the joint probability that (i) the primary processor fails prior to completing the checkpoint and (ii) the back-up processor completes the job starting from scratch, and the joint probability that (i) the primary processor completes the first checkpoint and (ii) proceeds according to an optimal $(k-1)$-checkpoint schedule on the remaining $\tau - x$ time units (see Fig. 2). Equation (2.1) is known as the *optimality equation*; together with $Q_0(\tau)$ in (1.1), it gives us in principle a method for solving for $Q_k(\tau)$, $k \geq 1$.

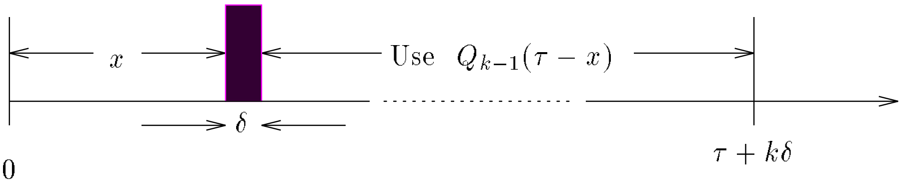
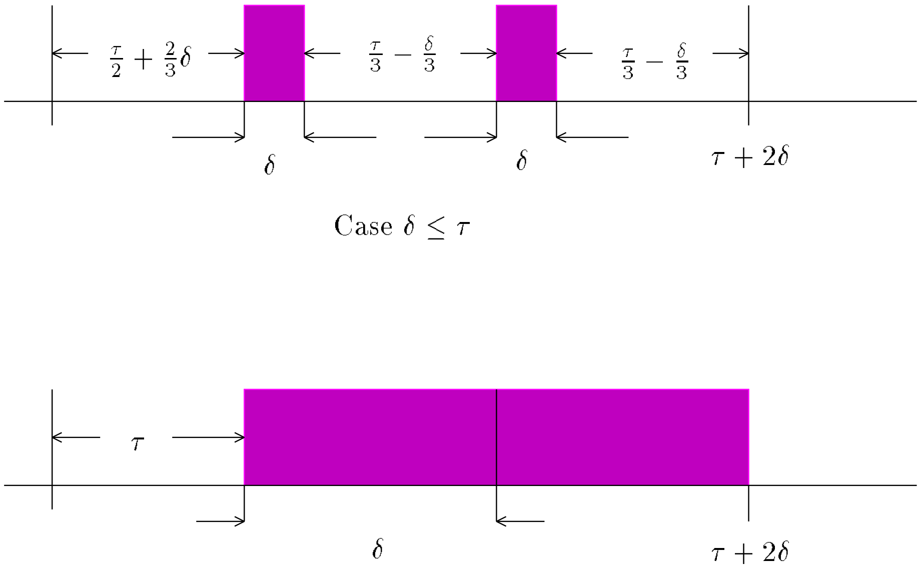


Figure 2: $Q_k(\tau)$

In the previous section we calculated $Q_1(\tau)$ and found that in the optimal 1-checkpoint schedule the primary processor, if it has not failed by $\tau/2$, attempts to make the checkpoint beginning at time $\tau/2$. We note for future reference that this result implies that, for all $k \geq 2$ as well, the last two checkpoint intervals are equal length in an optimal checkpoint schedule.

Let us carry the calculation one step further. Setting $k = 2$ in (2.1), differentiating the bracketed term with respect to x, setting the resultant expression equal to zero, and solving yields $x = \tau/3 + 2\delta/3$. The second derivative of the bracketed term in (2.1) is negative for all x and, consequently, $x = \tau/3 + 2\delta/3$ is the unique point at which the supremum occurs. However, the supremum in (2.1) is over x satisfying $0 \leq x \leq \tau$. Accordingly, for $\delta \leq \tau$ the supremum is achieved at $x = \tau/3 + 2\delta/3$ and for $\delta > \tau$ the supremum is achieved at $x = \tau$ (see Fig. 3). Therefore, if $\delta \leq \tau$,

$$Q_2(\tau) = (1 - e^{-(\tau/3 + 5\delta/3)})e^{-\tau} + e^{-(\tau/3+5\delta/3)} Q_1(2\tau/3 - 2\delta/3) \ ,$$



$\frac{\tau}{2} + \frac{2}{3}\delta$
$\frac{\tau}{3} - \frac{\delta}{3}$
$\frac{\tau}{3} - \frac{\delta}{3}$
δ
δ
$\tau + 2\delta$
τ
δ
$\tau + 2\delta$

Case $\delta \leq \tau$

Case $\delta > \tau$

Figure 3: $Q_2(\tau)$

and if $\delta > \tau \geq 0$,

$$Q_2(\tau) = (1 - e^{-(\tau+\delta)})e^{-\tau} + e^{-(\tau+\delta)}Q_1(0) \ .$$

Clearly, the checkpointing in Fig. 3 for the case $\tau < \delta$ serves no useful purpose. The optimal schedules for $k = 1, 2$ also illustrate the easily verified fact that, if $\delta \geq \tau$, then a schedule with no checkpoints has a maximum achievable completion probability.

The determination of the completion probability for optimal k-checkpoint schedules using (2.1) gets increasingly complex with increasing k. The following result is key to understanding the structure of optimal k-checkpoint schedules. Let I be a checkpoint interval. Then $||I||$ denotes the length of I.

Lemma 2.1. *Let I and J be consecutive checkpoint intervals in an optimal k-checkpoint schedule with I occurring before J and J not the last interval, i.e., the interval ending at $\tau + k\delta$. If $||I|| + ||J|| > \delta$ then $||I|| = ||J|| + \delta$ and if $||I|| + ||J|| \leq \delta$ then $||J|| = 0$.*

Proof. If $||I|| + ||J|| = 0$ the lemma is trivially true. Assume $||I|| + ||J|| = w > 0$. Let τ' denote the job processing time that remains at the start of checkpoint interval I. We will calculate the effect of moving the checkpoint ζ_J, the one between checkpoint intervals I and J, on the completion probability. Let x denote the distance from the beginning of

I to ζ_J. Since we are only interested in the effect on the completion probability of the location of checkpoint ζ_J as it is moved between ζ_I and ζ_K, the value of x ranges from 0 to w (see Fig. 4). Let $q(x)$ denote the completion probability of the schedule which is identical to the optimum k-checkpoint schedule except for the location of checkpoint ζ_J. We can write

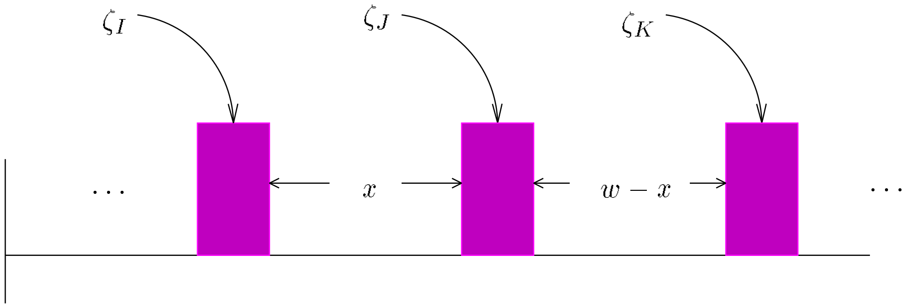


Figure 4: $q(x)$

$$q(x) = P + R[(1 - e^{-x-\delta})e^{-\tau'} + e^{-x-\delta}[(1 - e^{-w+x-\delta})e^{-\tau'+x} + e^{-w+x-\delta}S]] \ ,$$

where P is the joint probability that (i) the primary processor fails before reaching ζ_I (if I is the first checkpoint interval then starting the job immediately after ζ_I corresponds to starting the job from scratch) and (ii) the back-up processor successfully completes the job; R is the probability that the primary processor successfully completes checkpoint ζ_I; and S is the conditional completion probability of the schedule following checkpoint ζ_K given that ζ_K is successfully completed. The probabilities P, R, and S do not depend on x, the position of checkpoint ζ_J. Differentiating $q = q(x)$, we get

$$\frac{dq}{dx} = R[e^{-x-\delta-\tau'} - e^{-2\delta-w-\tau'+x}] \ .$$

Since $R > 0$, $\frac{dq}{dx} = 0$ if and only if $x = \frac{\delta+w}{2}$. Since $\frac{d^2q}{dx^2} < 0$ for all x, if $\frac{\delta+w}{2} < w$ then the maximum of $q(x)$ occurs at $x = \frac{\delta+w}{2}$; otherwise, $\frac{\delta+w}{2} \geq w$ and the maximum occurs at $x = w$. Accordingly, if $\delta < w$ we have that $||I|| = \frac{\delta+w}{2}$, $||J|| = w - \frac{\delta+w}{2} = \frac{w-\delta}{2}$, and $||I|| = ||J|| + \delta$. Otherwise, $||I|| = w$ and $||J|| = 0$. ■

Theorem 2.1. *Let I_j denote the jth checkpoint interval of an optimal k-checkpoint schedule, where I_j precedes I_{j+1}, $1 \leq j \leq k$. Either* *(i) there exists an integer b, $2 \le b \le k$, such that*

$$I_j = \begin{cases} a + (k-b+1-j)\delta, & 1 \le j \le k-b+1, \\ 0, \quad k-b+2 \le j \le k+1, \end{cases}$$

where

$$a \equiv \frac{1}{k-b+1}\left[\tau - \frac{(k-b)(k-b+1)}{2}\delta\right]$$

satisfies $0 < a \le \delta$*, or*

(ii) no such b exists and $\|I_j\| = \|I_{k+1}\| + (k-j)\delta$, $1 \le j \le k$, with

$$\|I_{k+1}\| = \frac{1}{k+1}\left[\tau - \frac{(k-1)k}{2}\delta\right] > 0 \ .$$

Proof. It is immediate from Lemma 2.1 that $\|I_1\|$, $\|I_2\|, \ldots$ is a nonincreasing sequence. Thus, all zero-length checkpoint intervals appear consecutively and at the end of an optimal k-checkpoint schedule.

Next assume that there is at least one zero-length checkpoint interval. Let b be equal to the number of zero-length checkpoint intervals (see Fig. 5). In this case b ranges from 2, when the last two checkpoint intervals are zero length, to k, when all except the first checkpoint interval are zero length. In the latter case the first checkpoint interval has length τ. Let a denote the length of the right-most nonzero checkpoint interval, I. We claim that a is less than or equal to δ. Otherwise, $\|I\| + \|J\| > \delta$, where J is the checkpoint interval immediately following I. By Lemma 2.1, the completion probability can be increased by moving the checkpoint between I and J to the left such that $\|I\| = \|J\| + \delta$. This is a contradiction and therefore $0 < a \le \delta$. By Lemma 2.1 the lengths of all of the nonzero length checkpoint intervals increase by δ as we move

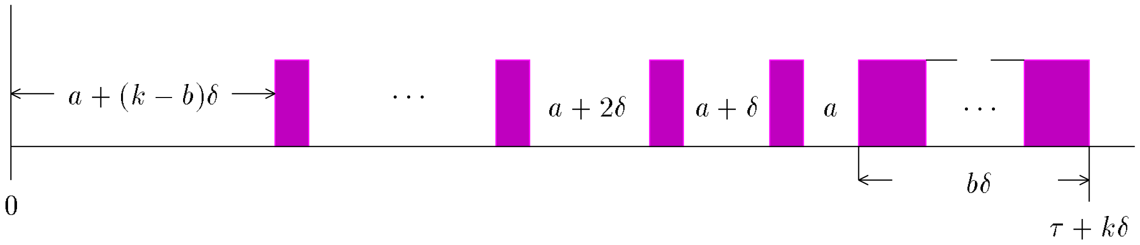


Figure 5: Case (i)

toward the beginning of the schedule. Summing the lengths of the checkpoint intervals and setting this equal to τ yields case (i) of the theorem.

Next we assume there are no zero-length checkpoint intervals. As noted earlier, since the last two checkpoint intervals and the last checkpoint form an optimal 1-checkpoint schedule, the last two checkpoint intervals have the same length, $a = \|I_{k+1}\| > 0$ (see Fig. 6). The length of each of the remaining checkpoint intervals increases by δ as we move toward the beginning of the schedule. Summing the lengths of the checkpoint intervals and setting the sum equal to τ, we get case (ii) of the theorem. ■

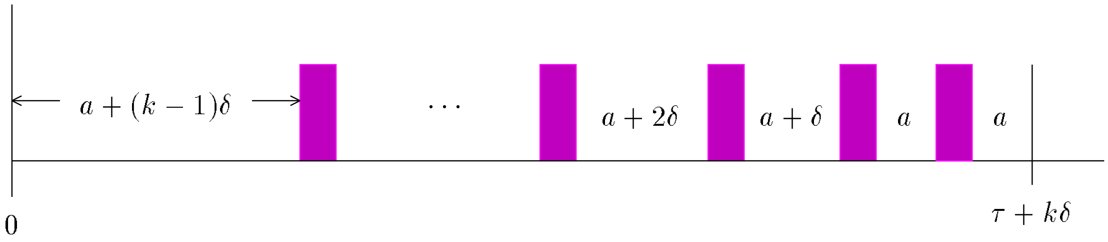


Figure 6: Case (ii)

The next result gives an explicit formula for the maximal completion probability for any fixed $k \geq 0$.

Theorem 2.2. *Let $k \geq 0$, $\tau > 0$, and $2\tau/\delta > k(k-1)$. Then*

$$Q_k(\tau) = e^{-(\tau+k\delta)} + e^{-\tau}\left(\frac{1-e^{-(k+1)\delta}}{1-e^{-\delta}}\right) - (k+1)e^{-\left(\frac{k+2}{k+1}\tau+\frac{k(k+3)}{2(k+1)}\delta\right)} . \qquad (2.2)$$

Proof. The proof is by induction on k. The case $k = 0$ is easy to verify (see (1.1)). Assume $k > 0$ and that the theorem holds for all nonnegative values smaller than k. Since we have by assumption that $2\tau/\delta > k(k-1)$, case (ii) in Theorem 2.1 holds. This means that the lengths of all the checkpoint intervals in an optimal k-checkpoint schedule are positive and, using Theorem 2.1, we can compute x_1, the length of the first checkpoint interval, *viz.*, $x_1 = \frac{\tau}{k+1} + \frac{k^2+k-2}{2(k+1)}\delta$. Since x_1 is the point at which the supremum is attained in (2.1), if we substitute x_1 for x in the bracketed term and use the induction hypothesis for $Q_{k-1}(\tau - x_1)$, the theorem follows. ■

3. Finding the Optimum Number of Checkpoints

Define the optimal completion probability to be $Q(\tau) = \max_{k \geq 0} Q_k(\tau)$, and define k_{opt} to be the smallest value of k such that $Q(\tau) = Q_{k_{opt}}(\tau)$. It follows from Theorem 2.1, that the optimal k-checkpoint schedules of interest are those for which $k(k-1) < 2\tau/\delta$, for otherwise the completion probability could be increased by eliminating a checkpoint. Thus, a search for k_{opt} can be confined to the nonnegative integers smaller than $(1 + \sqrt{1 + 8\tau/\delta})/2$. Below we will narrow this search significantly. First, consider the case $k_{opt} = 0$.

Theorem 3.1. *We have* $k_{opt} = 0$ *if and only if* $\delta \geq \ln \frac{2}{1+e^{-\tau/2}}$.

Proof. If $k_{opt} = 0$ then $Q(\tau) = Q_0(\tau) \geq Q_1(t)$. But then $\delta \geq \ln \frac{2}{1+e^{-\tau/2}}$ from (1.4). It remains to show that if $\delta \geq \ln \frac{2}{1+e^{-\tau/2}}$, then $Q_0(\tau) \geq Q_k(\tau)$ for all $k \geq 1$. For $k = 1$ this is immediate from (1.4), so k_{opt} can not be 1. Suppose $k_{opt} \geq 2$, and let τ' denote the remaining processing time of the job after the next-to-the-last checkpoint in the optimum schedule. Since $\tau' < \tau$, we have $\delta \geq \ln \frac{2}{1+e^{-\tau/2}} > \ln \frac{2}{1+e^{-\tau'/2}}$. It follows from (1.4) that we can improve this alleged optimum schedule by removing the last checkpoint. This contradiction proves that $k_{opt} = 0$ must hold. ■

Notice that $\ln \frac{2}{1+e^{-\tau/2}} \leq \ln 2$ holds for all $\tau \geq 0$. Therefore we have the following condition, *independent of* τ, that implies the optimality of zero checkpoints.

Corollary 3.1. *If* $\delta \geq \ln 2$ *then zero checkpoints is optimum.*

By Corollary 3.1, we need only consider δ in $(0, \ln 2)$.

Theorem 3.2. *Let* $0 < \delta < \ln 2$. *If* $\delta \geq \tau$ *then* $Q(\tau) = Q_0(\tau)$. *Otherwise,* $\delta < \tau$ *and* $Q(\tau) = \min_{\lceil L \rceil \leq k \leq \lfloor H \rfloor} Q_k(\tau)$, *where*

$$\begin{aligned} H &= -1/2 + \sqrt{\frac{2\tau}{\delta} - \frac{7}{4}}, \\ L &= \frac{1}{2} - \beta + \sqrt{\frac{2\tau}{\delta} + \beta^2 + \frac{1}{4} - 3\beta} \end{aligned}$$

with $\beta = \frac{2}{\delta} \ln \frac{e^\delta}{2-e^\delta}$.

Proof. Since it never pays to make a checkpoint if $\delta \geq \tau$, assume $\delta < \tau$. By Theorem 2.1, we can restrict our minimization to nonnegative values of k satisfying $k(k-1) < 2\tau/\delta$.

This inequality can be strengthened by noticing that if the duration of the last checkpoint interval of an optimal k-checkpoint schedule is less than or equal to δ then the completion probability can be increased by eliminating the last checkpoint. Accordingly, we have $||I_{k+1}|| > \delta$ (Theorem 2.1, case (ii)). Simplifying this inequality, we get $k^2+k+2 < 2\tau/\delta$. Since $\tau > \delta$, $k^2+k+2-\frac{2\tau}{\delta} = 0$ has a positive and a negative root. Therefore k is bounded above by $\lfloor H \rfloor$, the floor of the positive root.

By (1.4), the remaining processing time after the last checkpoint in an optimal k-checkpoint schedule must not be too large for otherwise we could add a checkpoint and increase the completion probability; in particular, we must have $\delta \geq \ln \frac{2}{1+e^{-\frac{||I_{k+1}||}{2}}}$. Using this relation to bound $||I_{k+1}||$ and applying Theorem 2.1 case (ii), we get $k^2+(2\beta-1)k+2\beta-\frac{2\tau}{\delta} \geq 0$. It is not difficult to show that β is an increasing function of δ in $(0, \ln 2)$ and is bounded below by 4 (note that $\lim_{\delta\to 0} \beta = 4$). It follows that $k^2+(2\beta-1)k+\beta-\frac{2\tau}{\delta} = 0$ has two real roots and k is bounded from below by $\lceil L \rceil$, the ceiling of the larger of the two roots. ■

Corollary 3.2. *Let* $0 < \delta < \ln 2$ *and* $\delta < \tau$. *Then* $\lfloor H \rfloor - \lceil L \rceil \leq \beta - 1$ *where* $\beta = \frac{2}{\delta} \ln \frac{e^\delta}{2-e^\delta}$.

Proof. By Theorem 3.2 we have

$$H - L = \beta - 1 + \sqrt{\frac{2\tau}{\delta} - 7/4} - \sqrt{\frac{2\tau}{\delta} + \beta^2 + \frac{1}{4} - 3\beta}\ .$$

Since $\lfloor H \rfloor - \lceil L \rceil \leq H - L$, the first radical is smaller than $\sqrt{2\tau/\delta}$, and the second radical is greater than $\sqrt{2\tau/\delta}$, the corollary follows. ■

Figure 7 below shows β as δ varies from 0 to $\ln 2 - 0.05$. Since $\beta(\ln 2 - 0.05) = 9.2\ldots$, the difference between $\lfloor H \rfloor$ and $\lceil L \rceil$ is at most 8 over almost all of the interval $(0, \ln 2)$ and for δ in $(0, 0.47)$ the difference is at most 4.

Corollary 3.3. $k_{opt} - \sqrt{2\tau/\delta} = O(1)$ *as* $\delta \to 0$.

Corollary 3.4. $\lim_{\delta\to 0} Q(\tau) = (\tau+1)e^{-\tau}$.

Proof. This follows from Theorem 2.2 and Corollary 3.3. Substitute $\sqrt{2\tau/\delta}$ for k in $Q_k(\tau)$ and evaluate the constant term of a Taylor series expansion at $\sqrt{\delta} = 0$. ■

Note that Corollary 3.4 justifies the limit claimed in (1.5).

Figure 7: $\beta = \frac{2}{\delta} \ln \frac{e^{\delta}}{2-e^{\delta}}$

4. The Conditional Expected Job Completion Time

In this section we restrict our attention to optimal k-checkpoint schedules where $k \leq \lfloor -1/2 + \sqrt{2\tau/\delta - 7/4} \rfloor$ (Theorem 3.2). Experiments suggest that, for small $\delta > 0$, the completion probability increases as the number of checkpoints increases to the optimum, k_{opt}. This increase is accompanied by an undesirable increase in the conditional expected time to complete the job, given that it completes. To study this trade-off, we compute the conditional expected completion times as follows.

Consider an optimal k-checkpoint schedule with checkpoint interval durations $x_i = \|I_i\|$ for $i = 1, \ldots, k+1$. We will assume a scheduling/checkpointing policy such that, if k checkpoints are ever successfully made, then both processors are used thereafter until either the job finishes or both processors fail. (If $k = 0$, then both processors start immediately.) It is easy to see that this will stochastically decrease conditional job completion time and will have no effect on the completion probability, $Q_k(\tau)$.

Let *failure interval* l be the l^{th} of the $k+2$ intervals spanning $\mathbf{R}_+$ that are defined by the k checkpoint completion times and the time $\tau + k\delta$; failure interval l has length $x_l + \delta$, $l = 1, \ldots, k$; failure interval $k+1$ has length x_{k+1}; and failure interval $k+2$ is $[\tau + k\delta, \infty)$ (each interval is taken to be closed on the left and open on the right). Let $\xi^{(k)}$ be the event that the job successfully completes, and let $\xi_l^{(k)}$ be the joint event that the failure on the primary processor is in failure interval l and the job successfully completes. We have $\xi^{(k)} = \bigcup_{1 \leq l \leq k+2} \xi_l^{(k)}$, so if $E[C^{(k)} | \xi^{(k)}]$ denotes the conditional expected completion time under an optimal k-checkpoint schedule, then

$$E[C^{(k)}|\xi^{(k)}] = \sum_{1\le l\le k+2} E[C^{(k)}|\xi_l^{(k)}]P(\xi_l^{(k)}|\xi^{(k)}) \ . \tag{4.1}$$

Given that a failure on the primary processor occurs in failure interval l, $1 \le l \le k+1$, the conditional distribution of the failure epoch is uniform over the interval. Then,

$$E[C^{(k)}|\xi_l^{(k)}] = \frac{x_l+\delta}{2} + (l-1)\delta + \tau, \quad 1 \le l \le k \ . \tag{4.2}$$

If all k checkpoints are made before a failure occurs on the primary processor, then since the two processors run in parallel after a successful $k^{\rm th}$ checkpoint, we have

$$E[C^{(k)}|\xi_{k+1}^{(k)}] = E[C^{(k)}|\xi_{k+2}^{(k)}] = \tau + k\delta \ . \tag{4.3}$$

Note that, in the case of no checkpoints, we obtain the minimum conditional completion time

$$E[C^{(0)}|\xi^{(0)}] = \tau \ , \tag{4.4}$$

a result that also holds if $\delta = 0$. It remains to compute the conditional probabilities, $P(\xi_l^{(k)}|\xi^{(k)})$.

We have $P(\xi^{(k)}) = Q_k(\tau)$. A routine calculation then shows that, for $k \ge 1$,

$$P(\xi_l^{(k)}|\xi^{(k)}) = \frac{e^{-(\tau+(l-1)\delta)}(1-e^{-(x_l+\delta)})}{Q_k(\tau)}, \quad 1 \le l \le k \tag{4.5}$$

$$P(\xi_{k+1}^{(k)}|\xi^{(k)}) = \frac{e^{-(\tau+k\delta)}(1-e^{-x_k})}{Q_k(\tau)} \ , \tag{4.6}$$

$$P(\xi_{k+2}^{(k)}|\xi^{(k)}) = \frac{e^{-(\tau+k\delta)}}{Q_k(\tau)} \ . \tag{4.7}$$

Substitute (4.2), (4.3), and (4.5)–(4.7) into (4.1), use $x_l = x_1 - (l-1)\delta$, and simplify to obtain

$$E[C^{(k)}|\xi^{(k)}] - \tau = \frac{e^{-\tau}}{Q_k(\tau)}\left[\sum_{1\le l\le k}\frac{1}{2}(x_1+l\delta)(e^{-(l-1)\delta}-e^{-(x_1+\delta)}) + k\delta e^{-k\delta}(2-e^{-x_k})\right] \tag{4.8}$$

for the increase in conditional expected completion time over the minimum achievable value τ. Working out the sum in (4.8) gives

$$\begin{aligned} E[C^{(k)}|\xi^{(k)}] - \tau = \frac{e^{-\tau}}{2Q_k(\tau)}\Bigg[& x_1\frac{1-e^{-k\delta}}{1-e^{-\delta}} - k\left(x_1 + \frac{k+1}{2}\delta\right)e^{-(x_1+\delta)} \\ & + \delta\frac{1-(k+1)e^{-k\delta}+ke^{-(k+1)\delta}}{(1-e^{-\delta})^2} \end{aligned} \tag{4.9}$$

with $x_1 = \frac{\tau}{k+1} + \frac{k^2+k-2}{2(k+1)}\delta$. Substitution of $k = k_{opt}$ yields an expression for $E[C|\xi] - \tau$, the increase in the conditional expected completion time for a checkpointing schedule that maximizes the completion probability.

An asymptotic bound on the increase in the conditional expected completion time as $\delta \rightarrow 0$ is available from earlier results. First, from (4.1) write

$$E[C^{(k)}|\xi^{(k)}] \leq \max_{1 \leq l \leq k+2} (E[C^{(k)}|\xi_l^{(k)}]) \leq \frac{x_1 + \delta}{2} + k\delta + \tau \ ,$$

since $x_1 \geq \cdots \geq x_k$. Then

$$E[C|\xi] - \tau \leq \frac{\widehat{x}_1 + \delta}{2} + k_{opt}\delta \ ,$$

where $\widehat{x}_1$ is the duration of the first checkpoint interval of the optimal k_{opt}-checkpoint schedule.

Corollary 3.3 then shows that, for fixed τ,

$$E[C|\xi] - \tau = O(\sqrt{\delta\tau}) \quad \text{as} \quad \delta \rightarrow 0 \ , \tag{4.10}$$

where the hidden multiplicative constant is independent of τ as well as δ.

In the figures below we plot $Q_k(0.2)$ and $E[C^{(k)}|\xi^{(k)}]$ as we increase k from 0 to k_{opt} for values of δ equal to 0.01, 0.001, and 0.0001. These plots were obtained using Mathematica and equations (2.2) and (4.9). Although the equations are meaningful only for integral values for k, for ease of rendering and because the equations can be evaluated for nonintegral k, we let k vary continuously from 0 to k_{opt}.

Figure 8: $Q_k(0.2)$ and $E[C^{(k)}|\xi^{(k)}]$ with $\delta = 0.01$

An obvious feature of these plots is that the curves for $Q_k(0.2)$ are relatively flat in the region approaching k_{opt} and the curves for $E[C^{(k)}|\xi^{(k)}]$ are increasing as k approaches k_{opt}. This suggests we can reduce the number of checkpoints below k_{opt} at the cost of only a slight decrease in the completion probability and at the same time decrease $E[C^{(k)}|\xi^{(k)}]$. In the figures we give the values of $Q_k(0.2)$ and $E[C^{(k)}|\xi^{(k)}]$ for $k = k_{opt}$ and one other value of k. For example, when $\delta = 0.001$, $k_{opt} = 18$. However, if we use $k = 4$, the completion probability decreases by 0.0014 and $E[C^{(k)}|\xi^{(k)}]$ decreases by 0.0107. This is a 0.14 percent decrease in the completion probability and a corresponding 5 percent decrease in the expected finishing time.

Figure 9: $Q_k(0.2)$ and $E[C^{(k)}|\xi^{(k)}]$ with $\delta = 0.001$

Figure 10: $Q_k(0.2)$ and $E[C^{(k)}|\xi^{(k)}]$ with $\delta = 0.0001$

5. Final Remarks

The results in this paper, particularly Theorems 2.1 and 3.2, provide the basis for an effective engineering solution to the optimal-checkpointing problem set in Section 1. However, there are interesting, unresolved theoretical issues. A prime example is the detailed functional dependence of the completion probability on the number k of checkpoints. One might expect that this probability would increase monotonically as k increases from 0 to k_{opt} and then decrease monotonically thereafter. Whether this is in fact true remains an open problem.

There are also many interesting generalizations and extensions of the original problem. For example, one could consider $m > 2$ processors, assume a more general failure law, have processors of different speeds, include more than one job, have a stochastic job completion time, or look at other criteria, such as maximizing the probability that the job completes by a given deadline. Most of these problems seem hard, if explicit results such as those of this paper are the objective. However, a discretization of the problem may lead to effective computational approaches. For example, consider the model of this paper extended to $m > 2$ processors, where even for three processors our formulation via stochastic dynamic programming is confounded by the lack of an explicit formula for the optimum number of checkpoints for two processors as a function of τ and δ. We describe below a discrete model and its salient results; because of space constraints, many details must be left to the full version of the paper.

For $m > 2$ processors, consider a discrete-time model in which all events and decisions take place at epochs which are uniformly spaced in time. Consider a single job with a known processing time $n > 0$ where n is an integer. Associated with each processor is a *time-to-failure* random variable which takes on positive integer values. The time-to-failure random variables are independent and geometrically distributed with the parameter $p = 1 - q$ denoting the probability that a processor fails in one time unit $(0 < p < 1)$. The checkpoint procedure takes a fixed amount of time denoted by δ where δ is a positive integer. If a processor fails the job may be resumed by some other processor beginning at the latest successful checkpoint. For convenience, we assume there is an initial checkpoint corresponding to zero accumulated processing time for the job. As before, the objective is to maximize the probability of completing a single job assuming a fixed number of processors.

A policy π is a rule that specifies the number of epochs until the next checkpoint, given the number m of available processors and the remaining job processing time, n. If $\pi(m, n) = 0$ then no checkpoint is to be made; otherwise, if $\pi(m, n) = k > 0$, then the next checkpoint is to be made after processing the job for k additional epochs. We assume that $\pi(1, n) = 0$, since there is no point in making a checkpoint if there are no backup processors.

Let $Q_\pi(m, n)$ denote the completion probability under π. Then

(i) $Q_\pi(m, 0) = 1$ for all $m \geq 1$.

(ii) $Q_\pi(1, n) = q^n$ for all $n \geq 1$.

(iii) Let $m \geq 2$ and $n \geq 1$. If $\pi(m, n) = 0$ then

$$Q_\pi(m, n) = q^n + (1 - q^n) Q_\pi(m - 1, n) \; .$$

Otherwise, $\pi(m, n) = k > 0$ and

$$Q_\pi(m, n) = (1 - q^{k+\delta}) Q_\pi(m - 1, n) + q^{k+\delta} Q_\pi(m, n - k) \; .$$

Let $Q(m, n) = \sup_\pi \{Q_\pi(m, n)\}$. It is easy to see that $Q(m, n)$ is determined by the following set of optimality equations:

(i) $Q(m, 0) = 1$ for all $m \geq 1$.

(ii) $Q(1,n) = q^n$ for all $n \geq 1$.

(iii) $Q(m,n) = \max[V, W]$ where

$$\begin{aligned} V &= q^n + (1-q^n)Q(m-1,n) \quad \text{and} \\ W &= \max_{1 \leq k < n-\delta} [(1-q^{k+\delta})Q(m-1,n) + q^{k+\delta}Q(m,n-k)] \ . \end{aligned}$$

The range for k in W does not include zero since it does not pay to begin with another checkpoint and k is never greater than or equal to $n-\delta$ since starting a checkpoint of length δ when there is no more than δ left to go on the job is never better than omitting the checkpoint.

The above specification of the function $Q(m,n)$ suggests an obvious computation with $O(m \cdot n^2)$ running time. The $O(n^2)$ component is contributed by the calculation of W. We have devised an improvement that computes the maximum in $O(\ln n)$ steps for all practical cases and thus the overall running time of our implementation is $O(m \cdot n \ln n)$. This estimate assumes unit costs for arithmetic operations, which is not the case if we use exact arithmetic.

Based on our calculations it appears that the number of checkpoints increases with the number of processors. This can have the effect of dramatically increasing the conditional expected job completion time. Trade-offs with the completion probability can be worked out in analogy with Section 4.

References

[1] B. Kalyanasundaram and K. R. Pruhs, Fault-Tolerant Scheduling (Extended Abstract), *Proceedings, Symp. Th. Comput.*, ACM Press, New York, 115–124, 1994.

[2] L. B. Boguslavsky, E. G. Coffman, Jr., E. N. Gilbert, and Alexander Y. Kreinin, Scheduling Checks and Saves, *ORSA Journal on Computing*, Vol. 4, No. 1, Winter 1992.

[3] S. M. Ross, Introduction to Stochastic Dynamic Programming, Academic Press, 1983.

[4] E. G. Coffman, Jr. and E. N. Gilbert, Optimal strategies for Scheduling Saves and Preventive Maintenance, *IEEE Transactions on Reliability*, 39, 9–18, 1990.

[5] A. Duda, The Effects of Checkpointing on Program Execution Time, *Information Processing Letters*, 16, 221–229, 1983.

[6] V. G. Kulkarni, V. F. Nicola, and K. S. Trivedi, Effects of Checkpointing and Queueing on Program Performance, *Commun. Statist.-Stochastic Models*, 6(4), 615–648, 1990.

[7] P. L'Ecuyer and J. Malenfant, Computing Optimal Checkpointing Strategies for Rollback and Recovery Systems, *IEEE Transactions on Computers*, 37(4), 491–496, April 1988.

[8] D. Sleator and R. Tarjan, Amortized Efficiency of List Update and Paging Rules, *Communications of the ACM*, 28, 202–208, 1985.

[9] I. S. Gradshteyn and I. M. Ryzhik, *Table of Integrals, Series, and Products*, Fifth Edition, Academic Press, 1994.

[10] R. Geist, R. Reynolds, and J. Westall, Selection of a Checkpoint Interval in a Critical-Task Environment, *IEEE Transactions on Reliability*, 37(4), 395–400, October 1988.

[11] A. Goyal, V. Nicola, A. Tantawi, and K. Trivedi, Reliability of Systems with Limited Repairs, *IEEE Transactions on Reliability*, 36, 202–207, 1987.