Edgar H. Sibley
Panel Editor

Mean response time and availability as optimization criteria for checkpoint placement are better replaced by workable formulas that calculate the ratio between the marginal gain accrued to users who experience system failure and the (presumably slight) loss suffered on average by all users.

OPTIMIZATION CRITERIA FOR CHECKPOINT PLACEMENT

C. M. KRISHNA, KANG G. SHIN, and YANN-HANG LEE

Checkpointing is becoming increasingly popular in real-time and database systems as a means of mitigating the consequences of failure. Checkpointing involves storing authenticated process state information that can then be used in the event of failure to restart the affected computation from the latest checkpoint, instead of from the beginning. There are a number of articles in the literature dealing with the optimal number and placement of checkpoints [1–6, 9, 10]. Our purpose here is to address an important point that has been overlooked; that is, although the analyses presented are often ingenious and elegant, their practical validity is thrown into doubt by an inappropriate choice of optimization criteria. We propose alternative criteria that we believe to be of greater practical relevance.

In every published analysis of optimal checkpointing that we have seen, the only optimization criteria used are mean response time and/or availability. No explicit justification for this is ever given. A carelessness in choosing optimization criteria can result in inadequate and misleading performance analyses, for, in a real sense, optimization criteria are relative, not absolute. The simple act of choosing a particular criterion imposes a bias on the results that follow. By definition, optimization criteria specify the commodity that is of importance and therefore to be optimized. Optimization criteria have a very subtle influence on the way systems are viewed. In a sense, they are languages through which we seek to convey system performance. We know from experience with natural languages that these affect not only the way in which ideas are expressed but also the very ideas themselves. Optimization criteria are no exception. They manipulate our view of system behavior, contorting it to fit a prefabricated mold. For this reason, the choice of optimization criteria determines the practical usefulness of the results that are then derived.

Optimization criteria are performance measures. To choose them correctly, it is important to determine what it is we wish to express. In this case, where we wish to measure the advantages that accrue from checkpointing, it is best to trade the *benefits* derived from them against the *overhead* they impose: in other words, to carry out a *value analysis.* First, we shall show why mean response time and availability do not do so effectively and are therefore inadequate performance measures for checkpoints.

Mean response time has long been a favorite measure of computer performance among queuing analysts. It is much easier to compute than the higher moments of response time. That is, it is sometimes easy to obtain the mean response time even when the response time distribution is very difficult or impossible to calculate.

This work has been supported in part by NASA under Grant NAG 1-296. Any opinions here expressed are those of the authors and do not necessarily represent the views of NASA. All correspondence regarding this paper should be sent to Kang G. Shin.

Also, in many cases, the data that are supposed to be representative of the arrival or service time distributions are not known to a sufficient degree of accuracy to warrant obtaining the higher moments of the response time distribution. Again, the mean response time is often quite adequate for most day-to-day purposes.

Availability, which is defined as the percentage of time for which the system is operational, is another well-known measure. It is also basically a first-moment measure.

Our argument is that neither measure is suitable for checkpointing models. Although checkpoints are useful auxiliary means to enhance reliability, it must be assumed that the reliability of the system is already considerable *without* them. If the Mean Time Between Failures (MTBF) for a system were to be much less than, say, 50–100 hours, checkpointing would be only a minor concern; the designers would have much more pressing woes.

Nonetheless, when checkpointing models are studied using mean response time or availability as criteria, developers are frequently driven to the extremes of considering MTBF values of between 2 and 10 hours in their numerical examples. For systems that fail much less frequently than that (e.g., the systems that one comes across in practice), the improvements in both mean response time and availability due to checkpoints are too small to be numerically significant: Indeed, in many cases, checkpointing actually *increases* mean response time.

The reason for this is that both mean response time and availability express performance from the *system point of view*, and are therefore insensitive when it comes to probing the consequences of failure. Their chief use lies in characterizing the vast majority of tasks or transactions that do not experience system failure. They would be entirely appropriate for checkpoints if they were only devices to enhance *normal* (i.e., failure-free) operation. However, since checkpoints are meant only to improve the handling of failure, and indeed *add* overhead to the execution time of nonfailing transactions or tasks, we need to seek out more appropriate performance measures. In so doing, we assume that the failure rate is very low: less than 10^{-3} per hour. In order to be practically useful, any performance measures for checkpointing should have the following two basic features: They should express the resulting improvement in handling failures, and they should consider the impact on transactions or tasks that suffer no failure. Yet, at the same time, they should also not make unrealistic demands on system data. We shall consider the application of these principles first to special-purpose (i.e., real-time) systems and then to general-purpose systems.

REAL-TIME APPLICATIONS

Computers used in the control of critical systems whose malfunction may endanger life, public safety, or property have stringent reliability requirements. The chief distinction between the requirements for control computers as opposed to general-purpose computers is that, in the former, an outage of more than a very short duration may have catastrophic consequences. This is because real-time computers have *hard deadlines*, which when missed cause the controlled system to fail.

The function of checkpoints in real-time applications is to increase the probability of the system's recovering from failure quickly enough to meet hard deadlines. The overhead they impose is a slight increase in the response time of processes that do not suffer failure. This increase can degrade the quality of control provided to the controlled system and decrease system efficiency. The extent of this decline in efficiency can be used as a measure of the overhead imposed by the introduction of the checkpoints. In terms of a value analysis, the benefit that accrues from checkpoints is a reduction in the probability of missing hard deadlines (in [7] we call this the *probability of dynamic failure*), whereas the price that has to be paid is the decline of efficiency of the controlled system. (The decline in efficiency is expressed through a *cost function* [7] whose domain is the computer response time and whose range is the performance index—in units of energy, time, etc.—of the controlled system.) In connection with this decline in efficiency, we define *mean cost* as follows: Let $f(t)$ be the density function of the response time distribution for a given control task, and $g(t)$ the cost function associated with a response time of t for that task. Then, the mean cost accrued for every execution of the given task is equal to

$$MC = \int_0^\infty f(t)g(t)\,dt$$

Both the probability of missing a hard deadline and the overhead due to increased response time as reflected in the performance of the controlled system are of direct physical relevance insofar as they link the behavior of the controlling computer with that of the controlled system. By using these quantities, we are explicitly carrying out an analysis of the impact of the computer on the application. This was *not* the case with mean response time or availability, where the impact on the application is not made evident.[1] For this reason, the proposed measures have much greater physical meaning than do mean response time and availability.

Our value analysis is therefore a trade-off between the decline in the probability of missing a hard deadline and the possible increase in the average overhead incurred by the controlled system due to the added delay owing to checkpointing. We illustrate this by the following example.

Numerical Example

Let us examine the benefits that might accrue when checkpointing is used in the controlling computer of a computer-controlled aircraft in the final stages of descent just prior to landing. (Such aircraft are expected

[1] In any case, mean response time and availability are very poor yardsticks in the real-time domain.

to be operational early in the next century.) A mathematical analysis can be found in [8]; here, we restrict ourselves to a concise description.

The computer task to be considered is the deflection control of the aircraft's elevator. The four state variables of interest are the altitude, descent rate, pitch angle, and pitch angle rate of the aircraft. Constraints are prescribed for the value of each of these quantities at touchdown. The region over which touchdown is to occur is also specified, and the optimal trajectory is given for all four state variables. The task of the controlling computer is to estimate the value of the state variables periodically (every 60 milliseconds) and to compute the optimal deflection of the elevator. Owing to the nonzero response time of the computer, the control provided is only suboptimal. The performance index that is appropriate in this case is a weighted sum of the squares of the deviation of each of the state variables from the optimal trajectory. The greater the response time of the computer, the greater the deviation of the state variables from the optimal trajectory and, therefore, the greater the overhead imposed by the computer on the controlled aircraft. The hard deadline over the final part of the descent is found in [8] to be 60 milliseconds. The cost function (i.e., the weighted sum of the squares of the deviations of the state variables as a function of the controller response time) was also derived in [8] and is reproduced here as Figure 1. Our goal here is to use these data in computing the optimal number of checkpoints.

Let the MTBF be 10,000 hours; let the occurrence of error be a Poisson process with rate $\lambda = 1/\text{MTBF}$; and let t_0, t_s, and t_{ov} be the time needed to set up rollback, restart, and one checkpoint. Let us assume also that the saved state may be contaminated with probability p_s, which means the system can be recovered using rollback with probability $p_b = 1 - p_s$ and has to restart with probability p_s. Clearly, $p_b = 0$ when the number of checkpoints, n, is zero. Let the nominal execution time be denoted by ξ. Then, the total execution time ξ_t is given by

$$\xi_t = \xi + nt_{ov} + t_{rec}$$

where n is the number of checkpoints inserted and t_{rec} is the time overhead used in recovery. t_{rec} is a random variable that depends on the probability of failure, p_b, and p_s:

$$t_{rec} = \begin{cases} 0 & \text{if no error occurs} \\ t_b + t_{roll} & \text{if error occurs and the version is recovered by rollback} \\ t_s + t_{start} & \text{if error occurs and the task has to restart} \end{cases}$$

where t_{roll} and t_{start} are the computation undone because of rollback and restart, respectively. Since the ratio of the execution time of any single task to the MTBF is of the order of 10^7, we may assume that the probability of a second failure occurring to the same task is negligible. Let the checkpoints be placed at equidistant intervals and let $t_{inv} = \xi/(n + 1)$ be the interval between successive checkpoints. The density function of t_{roll} and t_{start} is given by $f_{roll}(t) = \lambda e^{-\lambda t}/(1 - e^{-\lambda t_{inv}})$ for $t \in [0, t_{inv}]$ and $f_{start}(t) = \lambda e^{-\lambda t}/(1 - e^{-\lambda \xi})$ for $t \in [0, \xi]$, respectively. The density function of the total response time ξ_t can be easily obtained from the above equations. Three cases are considered for the nominal execution time of the deflection task: 20 ms, 30 ms, and 40 ms. For each, the probability of dynamic failure and the mean cost that ensues with checkpoints is computed. To express the marginal benefit accrued (in terms of the reduction probability of dynamic failure p_{dyn}) against the price paid (in terms of the increased mean cost MC, i.e., operating overhead), we use the following trade-off ratio:

$$\text{Trade-off ratio}(n) = \frac{p_{dyn} \text{ with } (n-1) \text{ checkpoints} - p_{dyn} \text{ with } n \text{ checkpoints}}{\text{MC with } n \text{ checkpoints} - \text{MC with } (n-1) \text{ checkpoints}}$$

The results are presented in Table I. When the nominal execution time is 20 milliseconds, all the checkpoints do is increase the overhead, that is, the mean cost. No discernible drop is noticed in the probability of dynamic failure when checkpoints are added. The marginal gain in reliability on adding checkpoints is therefore zero.

However, as nominal execution time increases, checkpointing begins to cause a noticeable decrease in the probability of dynamic failure. This is expressed through a positive trade-off ratio: $n = 1$ for a nominal execution time of 30 ms; $n = 1, 2$ for 40 ms; and $n = 1, 2, 3, 4, 5$ for 50 ms. These ratios show that a tangible

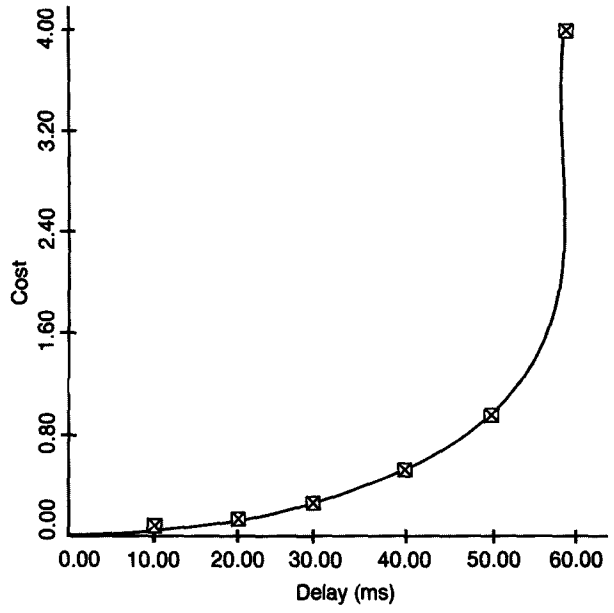

FIGURE 1. The Cost Function for a Real-Time System

TABLE I. Checkpoints in Real-Time Applications
(p_b = 0.9, MTBF = 10^4 hours, t_{ov} = 0.1 ms, t_b = 2.0 ms, t_s = 2.0 ms.)

n	Mean Cost	p_{dyn}	(Trade-off ratio) × 10^7
0	0.12848	0.3086E-15	—
1	0.12909	0.3086E-15	0.0
2	0.12971	0.3086E-15	0.0
3	0.13033	0.3086E-15	0.0
4	0.13095	0.3086E-15	0.0
5	0.13157	0.3086E-15	0.0

(a) Nominal execution time: 20 ms.

n	Mean Cost	p_{dyn}	(Trade-off ratio) × 10^7
0	0.26156	0.37037E-07	—
1	0.26431	0.37037E-08	121.5
2	0.26709	0.37037E-08	0.0
3	0.26991	0.37037E-08	0.0
4	0.27272	0.37037E-08	0.0
5	0.27567	0.37037E-08	0.0

(b) Nominal execution time: 30 ms.

n	Mean Cost	p_{dyn}	(Trade-off ratio) × 10^7
0	0.55352	0.30555E-06	—
1	0.55472	0.43055E-07	2177.0
2	0.55586	0.30555E-07	109.8
3	0.55694	0.30555E-07	0.0
4	0.55795	0.30555E-07	0.0
5	0.55891	0.30555E-07	0.0

(c) Nominal execution time: 40 ms.

n	Mean Cost	p_{dyn}	(Trade-off ratio) × 10^7
0	0.89694	0.46666E-06	—
1	0.90848	0.35666E-06	95.6
2	0.92025	0.26166E-06	80.6
3	0.93231	0.16666E-06	78.7
4	0.94466	0.71666E-07	76.9
5	0.95730	0.46666E-07	19.8

(d) Nominal execution time: 50 ms.

gain in reliability has been made for the indicated number of checkpoints (i.e., the probability of dynamic failure is reduced by a factor of 10 in each case). Of course, this has been achieved at the price of a certain increase in the mean cost, which is also reflected in the trade-off ratio.

Had mean response time and availability been used for the MTBF indicated, the results would have led to the recommendation that there be no checkpoints at all. The gain in reliability indicated above would have been masked by the high proportion of jobs that do not suffer failure. Mean response time is therefore a blunt instrument when it comes to probing the consequences of failure. A similar argument can be made for availability as a criterion.

GENERAL-PURPOSE SYSTEMS

The schema described above can be extended to encompass systems that do not have hard deadlines associated with executing tasks. The basic idea is to consider separately the impact of checkpoints on processes that do not experience failure and processes that do. If we are willing to countenance a performance vector in place of a scalar, the following might suffice:

$$p = \begin{bmatrix} MT_0 \\ MT_j \end{bmatrix}$$

where MT_0 = mean execution time for processes not experiencing failure, and MT_j = mean execution time for processes experiencing failures. This would be a much finer measure than unconditioned mean execution time or availability, while retaining all the advantages of only requiring the computation of the first moment of response time distribution. Of course, to compare two performance vectors, one would need a metric such as the following trade-off ratio:

$$\text{Trade-off ratio}(n) = \frac{MT_f \text{ with } (n-1) \text{ checkpoints} - MT_f \text{ with } n \text{ checkpoints}}{MT_0 \text{ with } n \text{ checkpoints} - MT_0 \text{ with } (n-1) \text{ checkpoints}}$$

The trade-off ratio computes the ratio of the marginal gain made to the mean execution time of jobs that experience failure to the marginal loss made to the mean execution time of jobs that are free from failure. Unlike the measure of mean response time, this measure allows for the fact that the execution of failure-free jobs may actually be degraded by the introduction of checkpoints.

The above trade-off ratio is used when we are interested in the role of checkpoints in reducing the execution time for those processes that undergo one or more failures, and not in what happens to processes that suffer n failures for some n. In other words, we average in the above trade-off ratio all fail-and-recover tasks regardless of the number of failures; the number of failures is generally no more than one. If, for some reason, one wished to consider separately the handling of processes according to the number of failures they suffered, an expanded performance vector $P = [MT_0, MT_1, \ldots]^T$ results, and it is not obvious what the metric for P would be.[2]

A second useful measure is the effect of checkpoints on various percentiles of execution time. This is analogous to the p_{dyn} computation for real-time systems that we referred to earlier.

We now come to the second condition, namely, that the performance measures should not make unrealistic demands on the data that go into calculating them. In computing response time distribution, which would normally be arduous and sometimes impossible, the assumption of low failure rate comes to our rescue. Since the failure rate is assumed to be small, the distribution of failure between two adjacent checkpoints can quite accurately be taken to be uniform.

To convince the reader that these measures are indeed computationally feasible and practical, a numerical example is presented below.

[2] We need a metric for comparing two vectors.

Numerical Results

Some numerical results for general-purpose systems are presented in Table II. We see that although unconditioned mean execution time increases in all cases, the mean execution time taken over all jobs that experience failure is reduced with the introduction of checkpoints until more than six checkpoints have been introduced in the first case (with nominal execution time 70 ms) and more than eight checkpoints in the second (with nominal execution time 90 ms). The reason why more checkpoints yield a benefit in the second (Table IIb) is that the greater nominal execution time increases the penalty incurred on a restart. Here again, if mean execution time had been chosen as a criterion, the optimal number of checkpoints recommended would have been zero. Of course, adding checkpoints increases the mean execution time taken over all jobs in the system but, at the same time, markedly reduces the execution time for jobs that suffer some failure. Mean response time fails entirely to indicate what the trade-off ratio does; that is, it fails to show explicitly what is gained as opposed to what is lost.

As in the real-time case, mean response time would have masked the reduction in response time for jobs that suffer failure. Again, this particular limitation of mean response time can be carried over to availability.

TABLE II. Checkpoints in General-Purpose Systems
Checkpoint establishment overhead = 0.5 ms;
MTBF = 10^4 hours and no hard deadline.

Number of Checkpoints	MT_f	MT_o	Trade-off Ratio
0	105.8	70.0	—
1	92.3	70.5	26.99
2	88.1	71.0	8.33
3	86.3	71.5	3.67
4	85.4	72.0	1.80
5	84.9	72.5	0.86
6	84.8	73.0	0.33
7	84.8	73.5	−0.01
8	84.9	74.0	−0.22
9	85.1	74.5	−0.37
10	85.3	75.0	−0.49

(a) Nominal execution time: 70 ms.

Number of Checkpoints	MT_f	MT_o	Trade-off Ratio
0	135.8	90.0	—
1	118.3	90.5	34.99
2	112.8	91.0	11.00
3	110.3	91.5	5.00
4	108.9	92.0	2.60
5	108.3	92.5	1.40
6	107.9	93.0	0.71
7	107.8	93.5	0.28
8	107.8	94.0	0.00
9	107.9	94.5	−0.20
10	108.1	95.0	−0.35

(b) Nominal execution time: 90 ms.

SUMMARY

Our object in this paper has been to show that the measures of mean response time and availability that have conventionally been used to indicate the benefits that accrue from checkpointing are inadequate in that they are not sufficiently sensitive. To remedy this problem, we have proposed workable trade-off ratio formulas that calculate the price paid for checkpointing in terms of the decline in efficiency of the system as a whole. Our performance measure has been relatively simple: the ratio between the marginal gain accrued to users who suffer system failure and the (presumably slight) loss suffered on average by all users.

In this paper, no effort has been made to address the issue of user perception. At this stage, there is only a intuitive link between the above performance measure ratio and the response time as perceived by the user. More research remains to be done into the nature of user-perceived delays.

REFERENCES

1. Baccelli, F. Analysis of a service facility with periodic checkpointing. *Acta Inf. 15*, 1 (1981), 67–81.
2. Brodetskiy, G.L. Periodic dumping of intermediate results in systems with storage-destructive failures. *Eng. Cybern. 15*, 5 (Sept.–Oct. 1979), 685–689.
3. Chandy, K.M., Browne, J.C., Dissly, C.W., and Uhrig, W.R. Analytic models for rollback and recovery strategies in data base systems. *IEEE Trans. Softw. Eng. SE-1*, 1 (Mar. 1975), 100–110.
4. Chandy, K.M., and Ramamoorthy, C.V. Rollback and recovery strategies for computer programs. *IEEE Trans. Comput. C-21*, 6 (June 1972), 546–556.
5. Gelenbe, E. On the optimum checkpoint interval. *J. ACM 26*, 2 (Apr. 1979), 259–270.
6. Gelenbe, E., and Derochette, D. Performance of rollback recovery systems under intermittent failures. *Commun. ACM 21*, 6 (June 1978), 493–499.
7. Krishna, C.M., and Shin, K.G. Performance measures for real-time controllers. In *Performance 83*, A. Agrawala and S.K. Tripathi, Eds. North-Holland, Amsterdam, 1983, pp. 229–250.
8. Shin, K.G., Krishna, C.M., and Lee, Y.-H. Unified methods for evaluating real-time controllers: A case study. Computing Research Laboratory Rep. CRL-TR-23, The Univ. of Michigan, Ann Arbor, June 1983.
9. Tantawi, A.N., and Ruschitzka, M. Performance analysis of checkpointing strategies. In *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems* (Minneapolis, Minn., Aug. 29–31). ACM, New York, 1983, p. 129.
10. Young, J.W. A first order approximation to the optimum checkpoint interval. *Commun. ACM 17*, 9 (Sept. 1974), 530–531.

CR Categories and Subject Descriptors: D.4.5 [**Operating Systems**]: Reliability—*checkpoint/restart, fault-tolerance*; C.3 [**Special-Purpose and Applications-Based Systems**]: *real-time systems*; C.4 [**Performance of Systems**]: *performance attributes; reliability, availability, and serviceability*
General Terms: Performance, Reliability

Received 8/83; revised 12/83; accepted 3/84

Authors' Present Address: C.M. Krishna, Kang G. Shin, and Yann-Hang Lee, Computer Research Laboratory, Dept. of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI 48109.