```
        DO 140 J=KP1,N
          SUM = 0.0
          DO 130 I=1,KM1
            SUM = SUM + D(K,I)*D(I,J)
  130     CONTINUE
          D(K,J) = -SUM + D(K,J)
  140   CONTINUE
  150 CONTINUE
      LAST = L - M2
      IF (LAST.EQ.0) GO TO 190
      K = M2
      M2P1 = M2 + 1
      DO 180 I=1,LAST
        K = K + 1
        DO 170 J=M2P1,N
          SUM = 0.0
          DO 160 II=1,M2
            SUM = SUM + D(K,II)*D(II,J)
  160     CONTINUE
          D(K,J) = -SUM + D(K,J)
  170   CONTINUE
  180 CONTINUE
  190 RETURN
      END
      FUNCTION GVAL(T, ID, N, M, X, C)
C INPUT T,ID,N,M,X,C
C THE PARAMETERS N,M,X,C ARE AS IN GSF  AND
C COMPLETELY DESCRIBE THE G-SPLINE.
C T IS A REAL NUMBER AND ID A POSITIVE INTEGER.
C GVAL PRODUCES THE ID-1 ST DERIVATIVE OF THE SPLINE
C AT T. GVAL AUTOMATICALLY PRODUCES 0 IF ID.GT.M*2
      DIMENSION X(100), C(8,100), S(8)
      IORD = 2*M
      IF (ID.GT.IORD) GO TO 130
C BINARY SEARCH FOR KNOT SUCH THAT
C X(KNOT-1).LT.T(KNOT)
      KNOT = 1
      IF (T-X(KNOT)) 70, 70, 10
   10 KNOT = N
      IF (T-X(KNOT)) 20, 60, 60
   20 KUP = N
      KLO = 1
   30 IF ((KUP-KLO).EQ.1) GO TO 70
      KNOT = (KUP+KLO)/2
      IF (T-X(KNOT)) 50, 70, 40
   40 KLO = KNOT
      KNOT = KUP
      GO TO 30
   50 KUP = KNOT
      GO TO 30
C EVALUATION OF THE SPLINE
   60 IORD = M
      IF (ID.GT.IORD) GO TO 130
   70 Y = T - X(KNOT)
      IORD1 = IORD + 1
C SET UP SPLINE COEFFICIENTS
      DO 80 I=1,IORD
        MU = IORD1 - I
        S(I) = C(MU,KNOT)
   80 CONTINUE
C HORNERS SCHEME
      DO 100 K=1,ID
        IORD = IORD1 - K
        DO 90 I=2,IORD
          S(I) = S(I-1)*Y + S(I)
   90   CONTINUE
  100 CONTINUE
      FACT = 1.0
      IF (ID.EQ.1) GO TO 120
      IDM1 = ID - 1
      DO 110 I=1,IDM1
        FACT = FACT*FLOAT(I)
  110 CONTINUE
  120 GVAL = S(IORD)*FACT
      RETURN
  130 GVAL = 0.0
      RETURN
      END
```



# A First Order Approximation to the Optimum Checkpoint Interval

John W. Young
*Martin Marietta Corporation*

**Key Words and Phrases: checkpoint, job failure, operations, programming checkpoint, random failure, operations, programming**
**CR Categories: 2.43, 4.0**

To avoid having to restart a job from the beginning in case of random failure, it is standard practice to save periodically sufficient information to enable the job to be restarted at the previous point at which information was saved. Such points are referred to as checkpoints, and the saving of such information at these points is called checkpointing [1].

IBM OS (Operating System) includes a "Checkpoint/Restart" feature for performing checkpointing and restarting more or less automatically. Checkpointing may also be accomplished by cataloging appropriate data sets at the end of some number of job steps. Restarting at such points is then accomplished by providing appropriate JCL (Job Control Language) decks.

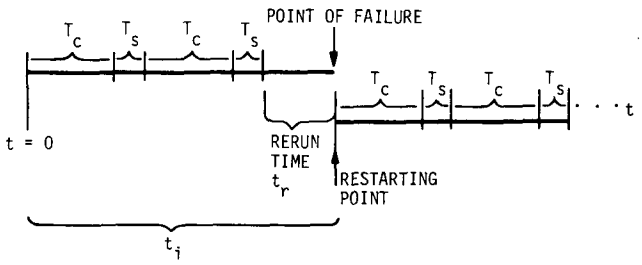
How frequently checkpointing should be done, however, is a very practical question that does not appear to have been addressed satisfactorily in the literature, and in practice a variety of rules of thumb with no substantial justification appear to be in use.

The purpose of this paper is to present a method for determining a first order approximation to the optimum time interval between checkpoints such as to minimize the time lost because of job failures due to random errors and, thereby, to minimize the cost of job failures due to random errors, assuming that computer use cost is essentially proportional to time in the machine. The execution time of a job may then be considered as a succession of the intervals $T_c$, the time interval between checkpoints, and $T_s$, the time to save information at a checkpoint taken alternately until a failure occurs. Such failure may occur during either an interval $T_c$ or $T_s$. In either case, execution of the program is resumed at the point in the program at which the previous checkpoint was successfully taken, assuming that a failure indication is observed at essentially the time a failure occurs, and the rerun or recovery time $t_r$ incurred due to the



Author's address: Computer Technology, Orlando Data Center, Data Systems Division, Martin Marietta Corporation, Orlando, FL 32805.

Fig. 1.




occurrence of such failure is then the time from the end of the previous interval $T_s$ to the point of failure, as shown on the time line in Figure 1. Note that the process of checkpointing begins anew after the occurrence of each failure.

Accordingly, when the length of the interval $t_i$ between failures lies between $n(T_c + T_s)$ and $(n + 1) \cdot (T_c + T_s)$, $n = 0, 1, \ldots$, then $t_i$ is composed of $n$ intervals of length $(T_c + T_s)$, plus the rerun time $t_r$. That is, $t_i = t_r + n\ (T_c + T_s)$. The time $t_l$ lost due to the occurrence of the failure consists of the time $n\ T_s$ taken to do the checkpointing prior to the occurrence of the failure, plus $t_r$. That is, $t_l = nT_s + t_r$.

From the previous expression this may be written as $t_l = t_i - nT_c$. In practice the occurrences of failures tend to cluster, because you think you have the cause of the failure fixed, but you don't. However, for our purposes we may take the occurrence of failures as essentially random (a Poisson process), with failure rate $\lambda$. Then the mean time $T_f$ between failures is $T_f = 1/\lambda$, and the density function $P(x)$ for the time interval of length $x$ between failures is given by $P(x) = \lambda e^{-\lambda x}$, which means that the probability that the interval between failures is of length $t_i$ is given by $\lambda e^{-\lambda t_i}\ \Delta t$, where $t < t_i < t + \Delta t$.

But the probability that the interval between failures is of length $t_i$ is precisely the probability that the lost time is of duration $t_l$. Therefore, denoting by $T_l$ the total time lost due to reruns caused by failures and to the time required for the process of checkpointing prior to the occurrence of failures, we have

$$T_l = \sum_{n=0}^{\infty} \int_{n(T_c+T_s)}^{(n+1)(T_c+T_s)} [t - nT_c](\lambda e^{-\lambda t})\, dt$$
$$= \lambda \int_0^{\infty} te^{-\lambda t}\, dt - \lambda T_c \sum_{n=1}^{\infty} n \int_{n(T_c+T_s)}^{(n+1)(T_c+T_s)} e^{-\lambda t}\, dt.$$

Integrating, we obtain

$$T_l = 1/\lambda + T_c \sum_{n=1}^{\infty} n\, [\exp(-\lambda(n+1)(T_c+T_s)) - \exp(-\lambda n(T_c+T_s))].$$

Factoring out the series, we obtain:

$$T_l = 1/\lambda - T_c\,(1 - \exp(-\lambda(T_c+T_s))) \cdot \exp(-\lambda(T_c+T_s)) \sum_{n=1}^{\infty} n \exp(-\lambda(n-1)(T_c+T_s)).$$

The series is in the form of the derivative with respect to $r$ of the geometric series with common ratio $r = \exp(-\lambda(T_c + T_s))$. Hence, its sum is $1/[1 - \exp(-\lambda(T_c + T_s))]^2$. Therefore,

$$T_l = 1/\lambda - T_c\,(1 - \exp(-\lambda(T_c+T_s))) \cdot \exp(-\lambda(T_c+T_s))/[1 - \exp(-\lambda(T_c+T_s))]^2,$$

which simplifies to

$$T_l = 1/\lambda + T_c/[1 - \exp(\lambda(T_c+T_s))].$$

To find the value of $T_c$ which minimizes $T_l$, we differentiate $T_l$ with respect to $T_c$, equate the result to zero, and solve for $T_c$:

$$\frac{dT_l}{dT_c} = \frac{1 - \exp(\lambda(T_c+T_s)) - T_c(-\exp(\lambda(T_c+T_s)))}{[1 - \exp(\lambda(T_c+T_s)]^2} = 0.$$

This means that $e^{\lambda T_c}\, e^{\lambda T_s}\, (1 - \lambda T_c) - 1 = 0$. Using the expansion of $e^{\lambda T_c}$ as far as the second degree term, the expression becomes: $1/2\ \lambda^2\ T_c^2 = 1 - e^{-\lambda T_s}$.

Since $T_f = 1/\lambda$, and in practice $T_s \ll T_f$, we may use the second order approximation to $e^{-\lambda T_s}$ to obtain $T_c^2 = 2T_sT_f - T_s^2$. Neglecting the term $T_s^2$ as being of second order with respect to $2T_sT_f$, we obtain $T_c = \sqrt{2T_sT_f}$.

This is an attractively simple result, and easy to apply in practice, as the following case illustrates.

It may be noted that following categories of causes of failure may occur randomly within a program and are therefore such that checkpointing is of value in minimizing the cost of failures due to such causes: Operator Error; Programming Error; Data Error; Hardware Failure; and System Failure.

If an insignificant number of errors of these categories are such as to cause concurrent failures in more than one partition or region, it can, therefore, be assumed that all such errors occur in the total running time of all partitions or regions in use.

Accordingly, for an IBM 360/50 installation, familiar to the writer, operating the equivalent of six days a week, of three shifts per day, namely 144 hours per week, using an average of two MFT partitions, and having a known failure rate of 19.5 errors per week, then $T_f = 2\ (144/19.5) = 14.72$ hours average time between failures.

The time required to catalog a data set was found to be about 7.5 sec, and the average number of data sets catalogued at a checkpoint is two. Then the average value of $T_s = (7.5)\ (2) = 15$ sec.

Using this data, the optimum value of $T_c$ is found to be $T_c = \left[2\,\frac{(15)}{60}\,14.72 \cdot (60)\right]^{\frac{1}{2}} = [441.6 = 21 \text{ min}]$.

It must be realized that the value of the optimum checkpoint interval is to be taken as a design goal which will not always be attained exactly, if checkpointing is being done only at the end of some number of steps which are of varying length.

Received January 1974

**References**

1. Jasper, David P. A discussion of checkpoint/restart. *Software Age* (Oct. 1969), 9–14.