Theoretical Computer Science 125 (1994) 131–147
Elsevier

Virus tests to maximize availability of software systems*

Erol Gelenbe
Department of Electrical Engineering, Duke University, Durham, NC, USA

Marisela Hernández
LAMIFA, Université de Picardie, Amiens, France

Abstract

Gelenbe, E. and M. Hernández, Virus tests to maximize availability of software systems, Theoretical Computer Science 125 (1994) 131–147.

Software systems in which many user's or programmers intervene may easily contain software items – such as viruses – which will endanger the integrity of the system. This paper proposes that in addition to the conventional recovery techniques, such as dumps and roll-back recovery, system availability be enhanced by the introduction of virus tests or other types of "failure tests". We present a model to analyze the effect of the failure rate, the frequency of virus and failure testing, and the frequency of periodic dumps, on global system availability. We assume that the "failure" rate of the system increases as time elapses beyond any individual instant at which a virus test or failure test has been carried out. Thus, we are dealing with a system in which failures will be naturally time-dependent. We compute the optimum value of the interval between dumps, and also the best time interval between virus or failure tests for this system. In order to illustrate the methodology of this work, numerical examples are presented for various time-dependent failure statistics.

1. Introduction

All software systems either large or small need to be protected from catastrophic failures, in case portions of the system architecture which support them fail. This is particularly true for large database systems in which several layers of recovery procedures are usually built into the software [10]. In such systems, variants of the

Correspondence to: M. Hernández, LAMIFA, Université de Picardie, 33 rue Saint-Leu, 80025 Amiens, France. Email addresses of the authors: erol@egr.duke.edu and marisela@math-info.univ-paris5.fr.

*This work was done at EHEI. Université de Paris V. Research supported in part by Programme C3-CNRS, Pôle Algorithmique Distribuée.

SSDI 0304-3975(93)E0119-O

checkpoint and roll-back recovery mechanism are used in order to enhance overall system reliability [15, 2].

The problem also arises in other systems which have a transaction-like behavior. For instance, in a large software development project in which modifications are carried out progressively and then validated periodically (very much as in a transaction oriented system in which updates are made and then validated) it may be necessary to introduce failure recovery mechanisms which protect the development activity from losses of information which may occur when there are system or subsystem failures.

Any such failure recovery mechanism is bound to induce additional costs and overhead, both during normal operation and during failure recovery. This is why much attention has been devoted to the performance of failure recovery mechanisms.

The earliest work in this area is that of [17], who has attempted to optimize the time between checkpoints in a database so as to obtain a satisfactory compromise between the time it takes to create a checkpoint, and the time it takes to recover from a failure. Indeed, the cost of checkpointing is high if checkpoints are frequent, but the advantage is then that in case of failure the recovery time will be shorter [3]. Other authors have analyzed the transaction queue length, or maximized system availability [7, 6], or considered response time as a primary measure of system performance [1], or examined the creation of checkpoints as a function of the number of transactions which occur between successive checkpoints [13]. These results have been extended to distributed systems [2, 8, 11], and to time-dependent failure rates [9]. The relationship between these models to queueing systems with service time interruptions has also been considered [12].

The purpose of this paper is to analyze a technique which we call "virus tests, or failure tests" for enhancing system availability. This is a very natural idea, since it is done in a routine manner in many systems. However, here, we assume that these "failure tests" will be carried out automatically at predetermined instants, much like checkpoints are carried out automatically under predetermined conditions in a database management system. It is assumed that the system operates with a dumping and roll-back recovery strategy for secondary memory failures. However, in addition, we imagine that certain database "failures" are in fact due to the introduction into the system of inconsistent or erroneous data, or more probably of some malicious code usually called a "virus", which at a later time may lead to the detection of a "failure" or to some other undesirable event.

Thus, we propose that in addition to the conventional recovery technique, the system availability be enhanced by the introduction of "failure tests" (or if you wish "virus tests"), which would be implemented as a set of procedures which examine the code, the data and/or the log of transactions which have been executed, in order to detect potential sources of failures or of inconsistencies.

After each failure test, one or more errors (or viruses, or failures, etc.) may or may not be detected in the system or data. If at least one failure or virus is detected, we

assume that the system has to go through a recovery procedure just as if there had been a system failure, using the most recent checkpoint. Otherwise transaction processing proceeds normally.

As a result of virus or failure testing, we expect the global system failure rate to behave in saw-tooth fashion as a function of time. Just after a failure test, possibly followed by a recovery procedure, the system failure rate drops to a low value because most of the potential sources of failures will have been detected in the failure test. The failure rate then increases as transaction processing proceeds and as we move away from the instant at which a virus or failure test was carried out. Thus, these tests can be viewed as a form of preventive maintenance carried out on the system's data, code, and past transactions.

The purpose of this procedure is clearly to improve system availability. Therefore, in the next section we shall present a model of the system in order to analyze the effect of the failure rate, the failure (or virus) tests, and the periodic dumps, on global system availability. We then compute the optimum value of the interval between dumps, and also the best time interval between failure tests for this system. The primary performance measure considered is the availability of the system for useful transaction processing. In the sequel we will indistinctly talk about failure tests or virus tests but it is clear that we are thinking primarily of all forms of testing, and specially about virus testing, which is aimed at removing malicious or nonmalicious sources of future failures or data inconsistencies.

Clearly, as the frequency of dumps (i.e. checkpoints) and of failure tests increases, so does the overhead related to these actions. However, the likelihood of failures and the subsequent costs related to failure recovery decrease. Therefore, it is important to determine the appropriate compromise which has to be made between these contradictory factors, so as to maximize system availability.

Thus, the computation of the optimum dump interval is presented in Section 3, while Section 4 is devoted to the computation of the optimum number of failure tests between dumps.

In Section 5 an algorithm is proposed to obtain the optimal dump interval and the number of failure tests between dumps, computational problems associated with the model, and numerical examples, are discussed as well.

The work presented in this paper can be viewed as a logical consequence of [6], where a roll-back recovery mechanism was analyzed and the problem of choosing an optimum checkpoint interval was formulated and solved, as well as of [9] where the mathematical tools for handling time-dependent failure rates in such models have been developed.

The results of the present paper are obtained as a function of certain parameters, such as the cost of creating dumps, the cost of making a failure test, the failure rate with the saw-tooth behavior described above or with constant or other time-dependent behavior. As far as the general methodology for availability optimization which we describe in this paper, these parameters can be chosen arbitrarily and introduced into the model.

2. The model

Let X_t be the state of the system at time t. X_t is given by

$$X_t = \begin{cases} 0 & \text{if the system is able to process transactions,} \\ 1 & \text{if it is recovering from a failure (possibly after a failure test),} \\ 2 & \text{if it is creating a dump,} \\ 3 & \text{if it is making a failure or virus test.} \end{cases}$$

We assume that at time 0 the system begins its functioning being able to process transactions, i.e. $X_t=0$ for $t \geqslant 0$ is constructed as follows:

(1) When the process enters state 2 it remains there for a random period independent of its past history, i.e. the time necessary for creating a dump, of general distribution function and with finite expected value ED. At the end of this time it returns to state 0.

(2) It can be assumed, with out loss of generality, that failures do not occur during dumping (state 2), during the failure or virus test (state 3) or during recovery (state 1).

(3) We denote by z_i the total time spent in take 0 (normal operation time) between the $(i-1)$th and ith failure tests, the 0th virus test is considered to be the most recent dump.

(4) It is assumed that the workload after a period of unavailability remains the same as before the interruption (dump, failure or virus test).

(5) Let Y_t (age) be the total time spent by the system in state 0 since the most recent dump preceding t:

$$Y_t = \int_{\hat{t}}^{t} 1(X_u=0)\,\mathrm{d}u, \quad \text{where } \hat{t}=\sup\{v: v<t \text{ and } X_v=2\}.$$

A transition from state 0 to state 1 occurs at some instant t due to the occurrence of a failure. Then the recovery time, i.e. the time spent in state 1 before the return to state 0, is given by a function $r(Y_t)$ which depends linearly on the age Y_t, where

$$r(Y_t)=\alpha Y_t+\beta$$

and $\alpha, \beta>0$ are constants. This formula can be justified as follows. After a failure, all the work done during time Y_t must be done again; this takes a time αY_t. The fixed time necessary to restart and reload the system being β.

(6) The instants of failure constitute a time-dependent Poisson process of parameter $\gamma(Y_t)$. The virus or failure tests have an effect on the time-dependent failure rate as shown in Fig. 1. A failure at time t will force the process X_t to go from state 0 to state 1.

(7) The system enters state 3 if a virus test is being made, and the time for making a virus test is a random variable of general distribution function and with finite expected value EV. At the end of the check with probability p an error is detected and the system enters state 1 for recovery, otherwise, it returns to state 0. The recovery

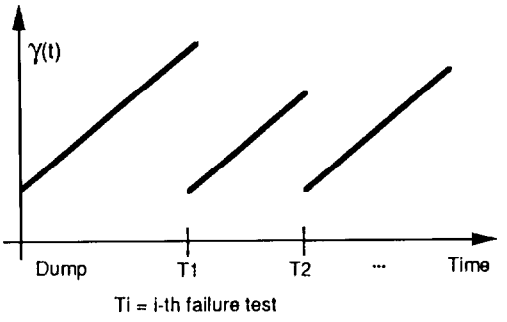

Fig. 1. The behavior of the failure rate $\gamma(t)$.

function $\hat{r}(Y_t)$ in this case will differ from $r(Y_t)$ in the fixed cost $\hat{\beta}$.

$$\hat{r}(Y_t)=\alpha Y_t+\hat{\beta}.$$

(8) The availability of the system is simply the proportion of time spent by the system in state 0. In steady state, it is the probability that the system is in state 0.

3. Interval between two successive dumps

We begin the analysis of the system presented in the previous section by considering an interval between two successive dumps, during which failure tests are made and failures may occur. In Fig. 2 we represent one such typical interval.

The average length of such an interval will be the sum of the total expected times spent in normal operation, in failure recovery, in carrying out the failure tests and creating the dumps. We shall examine each component of this duration, and compute its average value.

Let us first consider the total expected time spent in failure recovery between successive dumps. Clearly, it will be the expected time for recovery from failure test errors plus the expected recovery time for the time-dependent Poisson failures. We first compute the latter.

Assuming that there are n failures between the $(i-1)$th and ith failure tests, i.e. during z_i time units in normal operation, let $x_1<\cdots<x_n$ be the failure instants. As a consequence of the age-dependent Poisson assumption about failure instants, it is known (e.g. [5] p. 153) that for small enough interval of time δ_j, the conditional probability that failure j will occur in the small interval $[m_j, m_j+\delta_j]$ is given by

$$P(m_j\leqslant x_j<m_j+\delta_j,\ 1\leqslant j\leqslant n \mid n \text{ failures during } z_i \text{ time units in normal operation})$$

$$=\prod_{j=1}^{n}\left(\frac{\gamma(m_j)}{\Gamma(z_i)}\right)\delta_j,$$

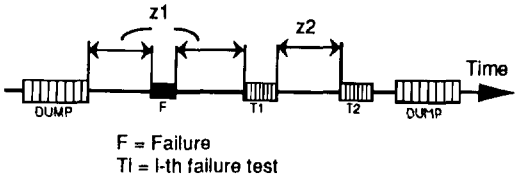

Fig. 2. The interval between dumps.

where

$$\Gamma(y)=\int_0^y \gamma(u)\,\mathrm{d}u.$$

The recovery time from the n failures occurred in the ith failure test interval, will then be

$$\int_{z_{i-1}}^{z_{i-1}+z_i} \mathrm{d}m_1 \ldots \mathrm{d}m_n \int_{z_{i-1}}^{z_{i-1}+z_i} \prod_{j=1}^{n} \left(\frac{\gamma(m_j)}{\Gamma(z_i)}\right) \sum_{h=1}^{n} r(z_1+\cdots+z_{i-1}+m_h)$$

$$= n \int_{z_{i-1}}^{z_{i-1}+z_i} \frac{\gamma(u)}{\Gamma(z_i)} r(z_1+\cdots+z_{i-1}+u)\,\mathrm{d}u,$$

because the cost function for recovery $r(\cdot)$ implies that all transactions which were processed by the system since the most recent dump have to be processed again.

Taking the expected value with respect to the number of failures, we obtain the following expression for the average recovery cost from random failures during z_i units of normal operation:

$$\sum_{n=0}^{\infty} n \frac{\mathrm{e}^{-\Gamma(z_i)}\Gamma(z_i)^n}{n!} \frac{1}{\Gamma(z_i)} \int_{z_{i-1}}^{z_{i-1}+z_i} \gamma(u) r(z_1+\cdots+z_{i-1}+u)\,\mathrm{d}u$$

$$= \int_{z_{i-1}}^{z_{i-1}+z_i} \gamma(u) r(z_1+\cdots+z_{i-1}+u)\,\mathrm{d}u.$$

If we assume that there are k failure tests in the dump interval, we have the following expression for the total expected time for recovery in between two successive dumps, where the first term corresponds to recovery due to errors detected during failure tests and the second one to failures arriving to the system.

$$Q_k(z_1,\ldots,z_k)=p\sum_{i=1}^{k}\int_0^{z_1+\cdots+z_i} \hat{r}(u)\,\mathrm{d}u+\sum_{i=1}^{k+1}\int_{z_{i-1}}^{z_{i-1}+z_i} \gamma(u) r(z_1+\cdots+z_{i-1}+u)\,\mathrm{d}u$$

and the total expected length of the interval between dumps, which we denote by $EI(k)$ is:

$$EI(k)=ED+k\,EV+\sum_{j=1}^{k+1} z_j+p\sum_{i=1}^{k}\int_0^{z_1+\cdots+z_i} \hat{r}(u)\,\mathrm{d}u$$

$$+\sum_{i=1}^{k+1}\int_{z_{i-1}}^{z_{i-1}+z_i} \gamma(u) r(z_1+\cdots+z_{i-1}+u)\,\mathrm{d}u$$

or

$$EI(k) = ED + k\,EV + \sum_{j=1}^{k+1} z_j + Q_k(z_1, \ldots, z_k).$$

This expectation allows us to compute the stationary probabilities (Π_j) of the process $\{X_t, t \geqslant 0\}$, applying the regenerative property [4] to the interval between dumps $EI(k)$:

$$\Pi_j \triangleq \lim_{t \to \infty} P[X_t = j], \quad j = 0, 1, 2, 3$$

and to compute them as follows:

$$\Pi_0 = \frac{\sum_{j=1}^{k+1} z_j}{EI(k)}, \qquad \Pi_1 = \frac{Q_k(z_1, \ldots, z_k)}{EI(k)},$$

$$\Pi_2 = \frac{ED}{EI(k)}, \qquad \Pi_3 = \frac{k\,EV}{EI(k)}.$$

Obviously, we also have the normalizing constraint:

$$\Pi_0 + \Pi_1 + \Pi_2 + \Pi_3 = 1$$

Indeed, it can be seen that $\{X_t, t \geqslant 0\}$ is a Markov renewal process, where the instants before or after each dump are the regeneration points of the process, and these formulae are a simple consequence of this fact.

3.1. Computation of the virus test intervals

In the previous section we computed the total average time between successive dumps, assuming that the length of each of the intervals between virus tests was given. In this section we turn our attention to the appropriate choice of the z_i, i.e. the time in normal operation between the $(i-1)$th and the ith virus test.

We shall make the following assumption which is based on a physical motivation: the total expected cost of each failure test in the same interval between dumps is kept constant. The "common sense" motivation behind this assumption is that during system operation, we would like that the availability of the system to normal transaction processing remain constant. Indeed, it would be rather annoying to a user that the quality of service received is very different depending on when the system is being accessed.

As a consequence of this assumption, we shall see that as we move away in time from the most recent dump, the successive virus tests must be carried out more frequently. Indeed, the expected cost of a virus test (which includes the cost of carrying out failure recovery) obviously increases as we move away from the most recent dump. This effect will be more or less important depending on the value of the probability of discovering an error during a virus test.

This assumption translates into the following general expression which relates the expected cost of the first failure test, to that of the ith one:

$$\int_0^{z_1} \gamma(u)r(u)\,du + p\int_0^{z_1} \hat{r}(u)\,du$$

$$= \alpha(z_1 + \cdots + z_{i-1}) \int_{z_{i-1}}^{z_{i-1}+z_i} \gamma(u)\,du + \int_0^{z_1+\cdots+z_i} \hat{r}(u)\,du$$

$$+ \int_{z_{i-1}}^{z_{i-1}+z_i} \gamma(u)r(u)\,du \quad \forall i = 2, \ldots, k. \tag{1}$$

In order to simplify the notation we write

$$\Gamma R(z_i) = \int_{z_{i-1}}^{z_{i-1}+z_i} \gamma(u)r(u)\,du,$$

$$\Gamma(z_i) = \int_0^{z_i} \gamma(u)\,du,$$

$$\hat{R}(z_i) = \int_0^{z_i} \hat{r}(u)\,du = \int_0^{z_i} (\alpha u + \hat{\beta})\,du$$

and (1) becomes

$$\Gamma R(z_1) + p\hat{R}(z_1) = \alpha(z_1 + \cdots + z_{i-1})\Gamma(z_i) + p\hat{R}(z_1 + \cdots + z_i) + \Gamma R(z_i)$$

$$\forall i = 2, \ldots, k, \tag{2}$$

which allows us to compute the different z_i as a function of z_1 with the constraint that each z_i must be positive.

Thus, we have now reduced the problem of computing the z_i to that of computing z_1. This will be carried out algorithmically in Section 5.

3.1.1. An example of a time-dependent failure rate

An often used time-dependent failure rate is the Weibull density [14] given by

$$\gamma(u) = \gamma\theta u^{\theta-1} + \gamma_0,$$

where $u \geqslant 0$ represents the time and γ, $\gamma_0 \geqslant 0$ and $\theta > 0$ are constants. Notice that this is a slight generalization of the usual Weibull failure rate [16] often used in reliability theory. When $\theta = 1$, we merely have a time-independent failure rate $\gamma + \gamma_0$ corresponding to a Poisson failure process. When $\theta = 2$ we have a linearly increasing failure rate; in general, the failure rate increases when $\theta > 1$. This failure process is thus a convenient representation of increasing, decreasing, or constant failure rates.

Substituting in (2) we obtain the following equation for z_i, the length of the ith interval between failure or virus tests, which can be obtained numerically from

$$\alpha S_1^{i-1}(\gamma(z_{i-1}+z_i)^\theta+\gamma_0 z_i-\gamma z_{i-1}^\theta)+\frac{p\alpha}{2}(S2_1^i)+p\hat{\beta}(S_1^i)+\frac{\gamma\alpha\theta}{\theta+1}(z_{i-1}+z_i)^{\theta+1}$$

$$+\frac{\gamma_0\alpha}{2}(2z_{i-1}z_i+z_i^2)+\gamma\beta(z_{i-1}+z_i)^\theta+\gamma_0\beta z_i-\frac{\gamma\alpha\theta}{\theta+1}(z_{i-1})^{\theta+1}-\gamma\beta z_{i-1}^\theta$$

$$-\frac{\gamma\alpha\theta}{\theta+1}z_1^{\theta+1}-\frac{p\alpha}{2}z_1^2-\hat{\beta}z_1=0, \tag{3}$$

where

$$S_1^{i-1}=\sum_{m=1}^{i-1} z_m,$$

$$S2_1^i=\sum_{m=1}^{i} z_m^2,$$

$$P_1^{i-1}=\sum_{m=1}^{i-1}\sum_{n=1}^{m-1} z_n z_m.$$

3.1.2. An example with the saw-tooth failure rate

In this case, $\gamma(u)$ has the behavior described in the following equations, which can be seen in Fig. 1.

$$\gamma(t)=\gamma_0+\gamma t, \qquad 0\leqslant t<z_1,$$
$$\gamma(t)=\gamma_0+\gamma(t-z_1), \qquad z_1\leqslant t<z_1+z_2,$$
$$\gamma(t)=\gamma_0+\gamma(t-\textstyle\sum_{j=1}^{i-1} z_j), \qquad \sum_{j=1}^{i-1} z_j\leqslant t<\sum_{j=1}^{i} z_j.$$

Clearly,

$$\Gamma(z_i)=\gamma\frac{z_i^2}{2}+\gamma_0 z_i,$$

$$\Gamma R(z_i)=\gamma\alpha\frac{z_i^3}{3}+(\gamma_0\alpha+\gamma\beta)\frac{z_i^2}{2}+\gamma_0\beta z_i.$$

Substituting in (2), we obtain the following equation for the interval z_i as a function of the $i-1$ preceding intervals:

$$\frac{z_i^3}{3}\gamma\alpha+\frac{z_i^2}{2}\gamma(\alpha S_1^{i-1}+\beta)+z_i^2\frac{\alpha}{2}(p+\gamma_0)+z_i(\alpha\gamma_0 S_1^{i-1}+p\hat{\beta}+\gamma_0\beta+p\alpha S_1^{i-1})$$

$$+\frac{p\alpha}{2}(S2_2^{i-1})+p\alpha P_1^{i-1}+\hat{\beta}(S_1^{i-1}-z_1)-\frac{z_1^3}{3}\gamma\alpha$$

$$-\gamma\beta\frac{z_i^2}{2}-\gamma_0\alpha\frac{z_1^2}{2}-z_1\gamma_0\beta=0. \tag{4}$$

4. Number of virus tests in the dump interval

In the previous sections we have

- obtained the system availability assuming that the z_i are known and that the number of failure tests k is fixed,
- derived a relationship between the successive z_i, assuming that z_1 is given.

Thus, we now have to develop a method to compute both k and z_1 in order to have a complete analysis of the system. These will be computed with the objective of maximizing system availability.

Let us turn to the problem of choosing k. We formulate it as follows.

After the $(i-1)$th failure test, a dump is carried out if the total expected cost associated with a new failure test is greater than the cost of making a dump.

More formally let k be the number of failure tests made in the interval between two successive dumps; we then choose

$$k = \inf\{i: Q_i(z_1, \ldots, z_i) \geqslant ED\}, \tag{5}$$

k will be determined numerically from the above expression, and will lead to the system availability Π_0 as a function of z_1:

$$\Pi_0 = \frac{\sum_{j=1}^{k+1} z_j}{ED + k\,EV + \sum_{j=1}^{k+1} z_j + Q_k(z_1, \ldots, z_k)}. \tag{6}$$

Finally, z_1 will be chosen so as to maximize Π_0 as shown in the next section.

5. Numerical implementation

We now turn to the numerical implementation of the above results.

The first problem we have to face is the computation of the first interval between the last dump and the first virus test (z_1), which then leads to the computation of the remaining z_i.

The second problem to be solved is the computation of the number (k) of failure tests in the interval between dumps satisfying (5).

Both of these computational problems will be solved using the algorithm given below.

We first choose an arbitrary z_1 and read the parameters of the model α, β, $\hat{\beta}$, γ, p, θ, ED, EV, where α is the proportion of transactions to be reprocessed if a failure has been detected, β the fixed reloading cost of recovery, $\hat{\beta}$ the fixed cost of recovery, γ the parameter of the failure rate, p the probability of finding an error during a virus test, θ the parameter of the Weibull failure rate, ED the expected time for making a dump, and EV is the expected time for making a failure test.

We have a formula to compute z_i as a function of $z_1, \ldots, z_{i-1}$ (see equations (3) or (4) depending on the shape of the failure rate). In other words, we can compute z_2 if we know z_1. We can compute z_3 using the values of z_1 and the computed z_2 and so on.

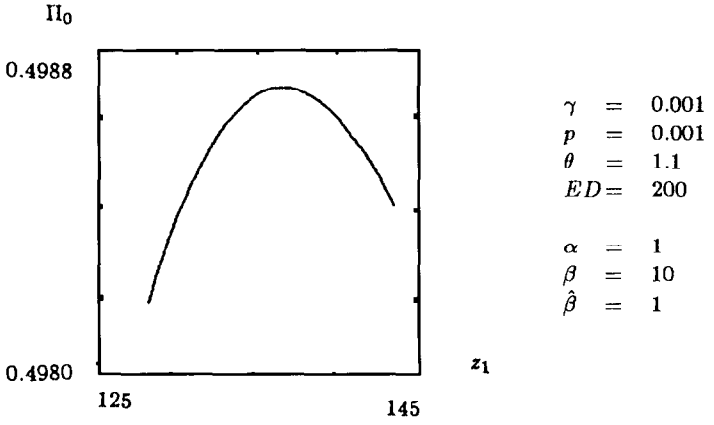

Fig. 3. Behavior of Π_0 as a function of z_1.

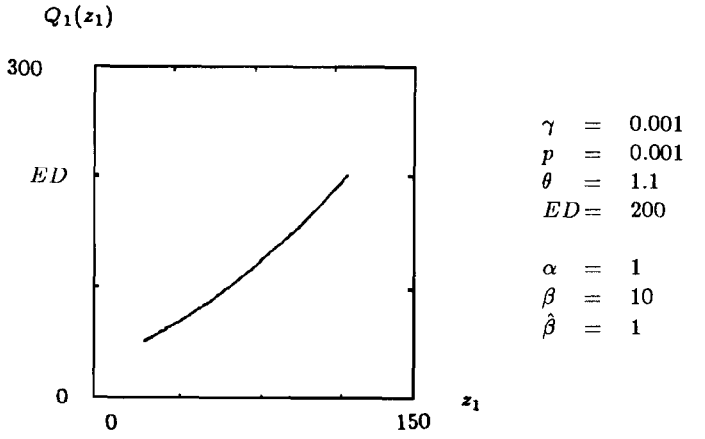

Fig. 4. Behavior of $Q_1(z_1)$ as a function of z_1.

This sequence of computations should be stopped as soon as the expected cost for reprocessing transactions is larger than or equal to the expected time ED for making a dump as indicated in (5).

Since the availability depends on a given value of z_1, let us call it $\Pi_0(z_1)$. We then determine numerically the value of z_1 maximizing Π_0, and call this optimal value z_1^*.

In Fig. 3, we present a numerical example of the dependence of $\Pi_0(z_1)$ on z_1.

Clearly, $Q_1(z_1)$ is an increasing function with values in $[0, ED]$. Then z_1^* will be bound by the condition

$$Q_1(z_1) \leqslant ED.$$

See Fig. 4 for an example of $Q_1(z_1)$.

z_1^* is found by carrying out a binary search in the interval $[L_I, L_S]$, where

$$L_I = 0 \quad \text{and} \quad L_S = \min\{x: Q_1(x) = ED\},$$

where the lower bound of z_1^* is clearly 0, while its upper bound is determined so that the expected total cost $Q_1(z_1)$ of the first failure test interval does not exceed ED.

5.1. Algorithm

In the following we detail the recursive procedure **binary**(L_I, L_S, MAX) for the binary search of the maximum of function Π_0 in the interval $[L_I, L_S]$ and the function $\Pi_0(z_1)$, which computes Π_0 for a given z_1. Note that in the case of a general Weibull rate we compute z_i using equation (3) and in the case of a saw-tooth failure rate we use equation (4).

Function $\Pi_0(z_1)$
  Read parameters α, β, $\hat{\beta}\gamma$, p, θ, ED, EV
    i = 1
    Repeat
      $i = i + 1$
      Compute z_i using equation (3)
    Until $Q_i(z_1, \ldots, z_i) \geqslant ED$ (equation (5))
  $k = i - 1$
  Compute Π_0 (equation (6))
  Send Π_0
Procedure binary (L_I, L_S, MAX)
  $L_M = \dfrac{L_I + L_S}{2}$; Left $\Pi_0 = \Pi_0(L_I)$; Right $\Pi_0 = \Pi_0(L_S)$
  If $L_I <> L_S + \varepsilon$
    then begin
      If Left Π_0 = Right Π_0
        then begin
          $L_{MI} = \dfrac{L_I + L_M}{2}$; $L_{MS} = \dfrac{L_M + L_S}{2}$
          Left $\Pi_0 = \Pi_0(L_{MI})$; Right $\Pi_0 = \Pi_0(L_{MS})$
            end
      If Left Π_0 < Right Π_0
        then binary (L_M, L_S, MAX)
        else binary (L_I, L_M, MAX)
        end
    else Send MAX = L_I
Algorithm Search-z_1^*
  $\varepsilon \leftarrow$ small; $L_I \leftarrow 0$
  Find $L_S = \min x$: $Q_1(x) = ED$
  binary (L_I, L_S, z_1^*)

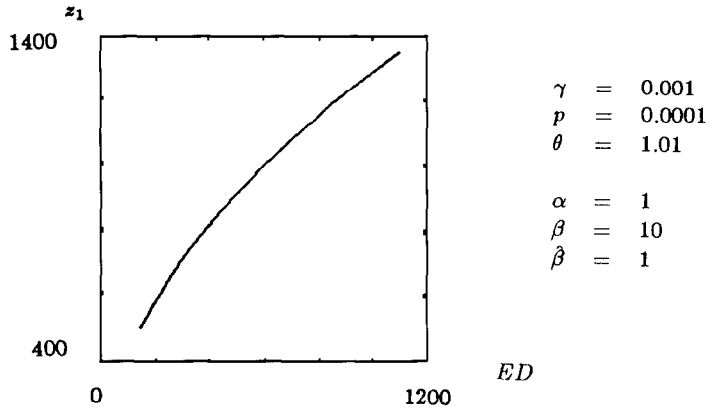

Fig. 5. z_1 as a function of ED (Weibull).

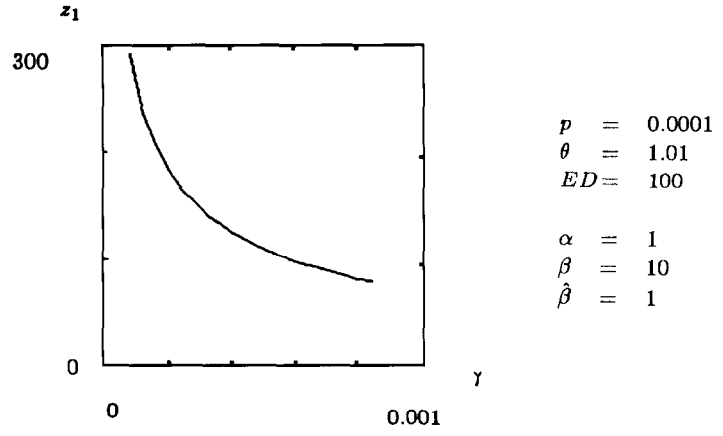

Fig. 6. z_1 as a function of γ (Weibull).

5.2. Numerical results for Weibull failure rate

The results are presented for the Weibull failure rate which has been widely used in reliability models. We first present some numerical examples concerning z_1^*, the total time in normal operation between the last dump and the first failure test. Then we turn to various numerical examples showing how system parameters affect system availability when z_1^* and k are chosen so as to maximize it.

With regard to z_1^* we notice the following effects:

- If the cost ED for making a dump increases then the interval z_1^* also increases (see Fig. 5)
- If the parameter γ of the failure rate increases (for the three cases constant, linear and Weibull) clearly the failure rate also increases and the first failure test has to be

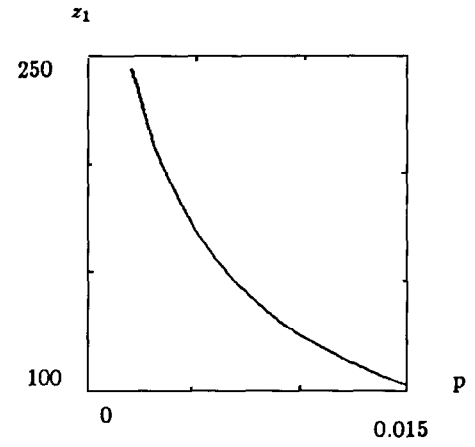

Fig. 7. z_1 as a function of p (Weibull).

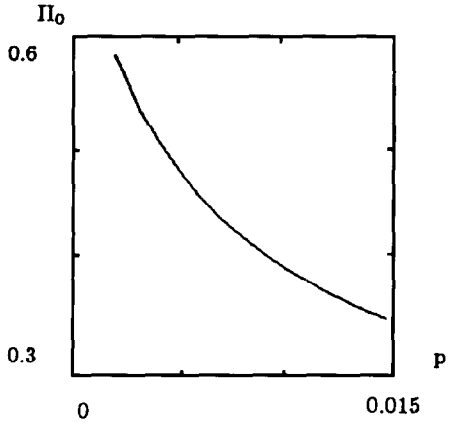

Fig. 8. Π_0 as a function of p (Weibull).

made earlier. Indeed, if the risk of failure is larger, it would be better to make a failure test earlier than if this risk is smaller (see Fig. 6)

- Similarly, if the probability p of discovering an error during a failure test is greater, then the interval z_1^* should be smaller (see Fig. 7)

Let us now turn to some results concerning system availability:

- If the probability p of finding a failure test error decreases then the availability will increase (see Fig. 8)
- If the cost ED of a dump increases, then the availability will obviously decrease (see Fig. 9)

The results were also verified for a saw-tooth failure rate.

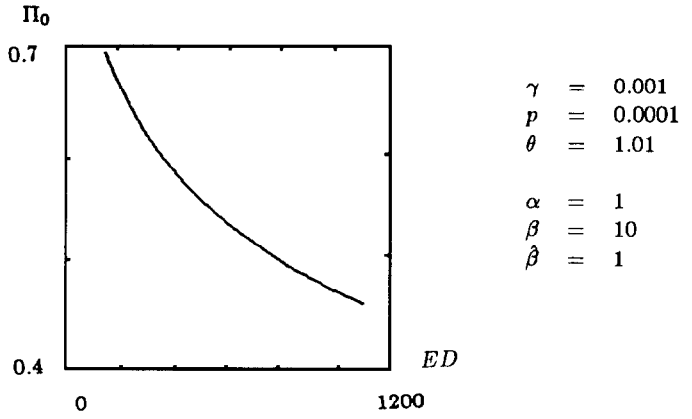

Fig. 9. Π_0 as a function of ED (Weibull).

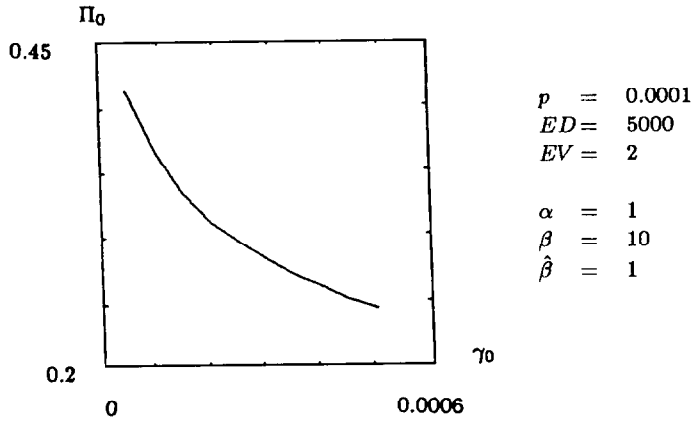

Fig. 10. Π_0 as a function of γ_0 (saw-tooth).

5.3. Numerical results for a saw-tooth failure rate

In Fig. 10 we consider the behavior of the availability Π_0 for a sawtooth-type time-dependent failure rate discussed in Section 3.1.2. This corresponds to the case where during a failure test, errors are detected with a certain probability p and a recovery procedure is carried out. Then the system failure rate drops to a low value as soon as the system begins its operation, but increases as time goes by and the system is used until a new failure test is carried out.

Π_0 is plotted against γ_0 for the saw-tooth failure rate. We observe that as γ_0 (the rate of increase of the failure rate) increases, the availability Π_0 decreases, as one may expect. Keeping the same model parameters ($\alpha = 1, \beta = 10, \hat{\beta} = 1, p = 0.0001, ED = 5000, EV = 2$) as in Fig. 10, we show the variation of Π_0 versus γ for the case of a linear

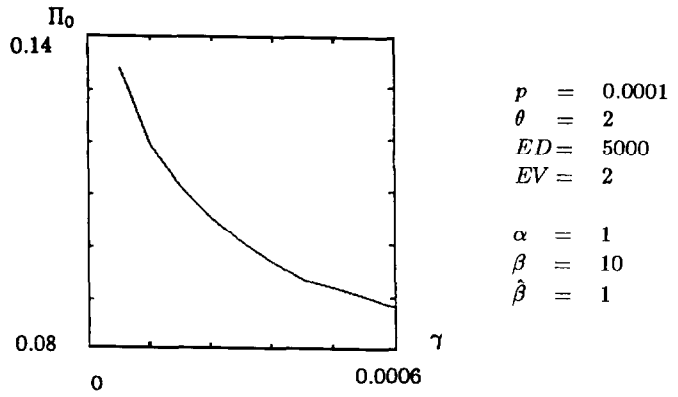

Fig. 11. Π_0 as a function of γ (Weibull).

Weibull failure rate (i.e. without the saw-tooth behavior) in Fig. 11. We see that the availability is appreciably greater in the saw-tooth case, showing that the failure tests are having the desired effect of improving system availability.

6. Conclusions

In this paper we propose the use of "failure tests" or "virus tests" in order to enhance the reliability and availability of a transaction oriented software system.

The basic idea is to carry out software tests and verifications at regular intervals concerning the data the system contains, and/or the transactions it has executed, in order to detect possible errors or inconsistencies. The purpose is to reduce the consequence such errors may have on system crashes, or to eliminate errors which would be detected later during normal system operation.

We discuss the appropriate choice of the number of failure tests and of the time between successive checks, assuming that the system is prone to failures. We expect that the system will also be equipped with a "roll-back recovery"-type mechanism in order to handle failures; this mechanism will be used for failure recovery both after a failure test and for random failures which occur during system operation. The basic time structure of system operation is established by the sequence of dumps, and virus or failure tests are carried between successive dumps.

A model is constructed in order to compute the best possible choice of intervals between dumps and failure test intervals so as to maximize system availability. This model is based on techniques initially developed in [6] for computing optimal checkpoint intervals, and in [9] which is devoted to time-dependent failure mechanisms and their analysis.

Numerical procedures for handling our model are developed and described, and examples are presented to illustrate the results we obtain.

It is hoped that this paper will contribute to the theory and practice of software systems having enhanced reliability properties.

References

[1] F. Baccelli, Analysis of a service facility with periodic checkpointing, *Acta Inform.* **15** (1981) 67–81.
[2] P. Bouchet, Procédures de reprise dans les systèmes de gestion de bases de données réparties, *Acta Inform.* **11** (1979) 305–340.
[3] K. Chandy, J. Browne, C. Dissly and W. Uhring, Analytic models for rollback and recovery strategies in database systems, *IEEE. Trans. Software Engng.* SE-**1** (1975) 100–110.
[4] E. Çinlar, *Introduction to Stochastic Processes* (Prentice-Hall, Englewood Cliffs, NJ, 1975).
[5] D. Cox, J. Miller, *The Theory of Stochastic Processes* (Methuen and Co Ltd., London and Colchester, 1965).
[6] E. Gelenbe, On the optimum checkpoint interval, *J. ACM* **26** (1979) 259–270.
[7] E. Gelenbe and D. Derochette, Performance of rollback recovery systems under intermittent failures, *Comm. ACM* **21** (1978) 493–499.
[8] E. Gelenbe, D. Finkel and S. Tripathi, Availability of a distributed computer system with failures, *Acta Inform.* **23** (1986) 643–655.
[9] E. Gelenbe and M. Hernández, Optimum checkpoints with age dependent failures, *Acta Inform.* **27** (1990) 519–531.
[10] T. Haerder and A. Reuter, Principles of transaction-oriented database recovery, *Comput. Surveys* **1** (1983) 287–317.
[11] R. Koo and S. Toueg, Checkpointing and rollback-recovery for distributed systems, *IEEE. Trans. Software Engng.* SE-**13** (1987).
[12] V. Nicola, A single server queue with mixed types of interruptions, *Acta Inform.* **23** (1986) 6465–486.
[13] V. Nicola and F. Kilstra, A model of checkpointing and recovery with a specified number of transactions between checkpoints, in: A.K. Agrawala and S. Tripathi, ed., *Performance '83* (North-Holland, Amsterdam, 1983) 83–99.
[14] S. Ross, *Stochastic Processes* (Wiley, New York, 1983).
[15] K. Salem and H. Garcia-Molina, Crash recovery mechanisms for main storage database systems, Research Report No. CS-TR-034-86, Dept. Computer Science, Princeton University, USA, 1986.
[16] K. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications* (Prentice-Hall, Englewood Cliffs, NJ, 1982).
[17] J. Young, A first order approximation to the checkpoint interval, *Comm. ACM* **17** (1974) 530–531.