ON THE PERFORMANCE OF SOFTWARE FAULT-TOLERANCE STRATEGIES*

A. Grnarov†, J. Arlat††, A. Avizienis

Computer Science Department
University of California
Los Angeles, California 90024, USA

ABSTRACT

In the paper a comparison of processing time and reliability performance for the Recovery Blocks scheme and N-Version Programming technique is presented. Derived queueing models can be useful in deciding which of the strategies should be used, depending on system parameters.

INTRODUCTION

Despite all efforts in careful design of software systems, they are unlikely to be error free and a lot of effort is still required to eliminate "bugs" after a system has been made available to the user. Recently self-checking software approaches3, 2 (limited to detection of software failures), and fault-tolerant software approaches3, 4 which allow to recover after a software failure occurred, were proposed. Figure 1 presents a general model for detection of software failures, and fault-tolerant software approaches3, 5 which allow to recover after a software failure.

Recovery Blocks scheme (RB) consists of three major entities: a primary alternate (A1), a list of supplementary alternates (A2, ..., AN–1) and an acceptance test (AT). The AT can be considered as an abstract implementation of the function performed by alternates. When conditions stated by AT are not met, a purging of data is performed and a new alternate is called. According to published work, in the paper a sequential application of RB is considered only. The N-Version Programming (NV) requires N ≥ 2 independently designed programs (versions) for the same function task. The obtained results after each stage of computation are compared and, in the case of disagreement, a preferred result can be identified by a majority vote (possibly exact) or another predetermined strategy (e.g., two-out-of-N). The bad versions are recovered by updating them with data provided by identified good versions and normal processing of all N versions can be resumed. The primary motivation for NV was the exploitation of fault-tolerant multiple hardware systems, which make a parallel execution of the versions (NVP) quite practical. However, in order to perform a fair comparison with RB, we also consider a "sequential" application of NV (NVS). In the model it is assumed that a program is partitioned into S segments, executed one at a given time, and that the "length" (execution time) of the segments is exponentially distributed. P1, P2, ..., PN represents alternates and versions in the case of RB and NV respectively. "Test" is the execution of AT in the case of RB, and the comparison of "c-vectors" for NV. "Continue" is the outcome of this state meaning that all actions that included as part of the "Repair" action, "Failure" is entered when a fault is undetected i.e., invalid results are passed on to the next segment. This can happen when correlated faults occur i.e., when in the same way fail: a) RB: an alternate and AT, b) NVS: two versions, c) NVP: majority of the versions.

In order to determine the average segment processing time (T) in the system with RB or NV fault-tolerance strategy, the queuing theory8 is used. The program segments are considered as "customers" and "service rates" of states are defined as follows: μi is the average service rate for processing a segment in a perfect system without fault-tolerance strategy; μR = μf / R is the average service rate of "Safe Down and Repair State" while "Continue" and "Failure" states have infinite service rates. Queueing models have been derived for each strategy, but due to the limitation of space, derivation of NVP model9 will not be presented in this paper.

The service rate of state (alternate) i is given by μi = 1 / ti, where the average alternate i processing time ti = tF + tR + tAT while tF is the average segment execution time, tR is the average alternate i setup time, and tAT is the average AT execution time.

For computational convenience, we use μi = μs(1 + aR + kR)−1 = μR(1 + bR)−1

* This research is supported in part by the ONR Contract N00014-79-C-0866, in part by the INRIA, France, in part by the NSF Contract MCS78-18918, and in part by a Grant of the Battelle Memorial Institute.
† On leave from Electrical Engineering Department, University of Skopje, Yugoslavia.
†† On leave from LAAS du CNRS, Toulouse, France.
where $\mu_s = \frac{1}{t_s}$, $a_s = \frac{t_a}{t_s}$, $k = \frac{t_k}{t_s}$ and $b_s = a_s + k$.

We also use the notation: $Q_i = c_i^s q_i$, where $c_i^s = 1$, and $c_i^p \in [1, \frac{1}{q_i}]$ for $j = 2, 3, \ldots, N-1$; and also $Q_i' = c_i^p q_i$, where $c_i^p \in [0,1]$ is the conditional probability $P(A_i, \text{AT fail in the same way if } A_i \text{failed})$.

The following assumption can be made for the model:

a) The setup times for different alternates do not differ significantly and accordingly, $a = a_s$, $b = b_s$, and $\mu_i = \mu_s(1 + b)^{-1} = \mu_s$.

b) The probabilities of no faults in the alternates are the same, i.e., $p_i = p_s$, $c_i^p = c_i^p$, and $c_i^p = c_i^p$.

c) Since the coverage of AT increases with its "length" (execution time) and since the probability of software faults increases with the length, we combine both of them in the probability of success of the acceptance test given by $P_{AT} = c^p q_i$ where the coverage $c = [(1-a)k+1]a$ and $P_{AT} = (1-k-q_i)$ while $\alpha$ is the minimal coverage and $q_i$ is the probability of software fault in AT if its length is the same as the length of an alternate.

According to the previous, and using the general series - parallel stages, the average segment processing time in the system is given by:

$$T_{RB} = \frac{1}{\mu_s} \left\{ 1 + Q_{AT} P' \left[ \frac{1 - (P')^{N-2}}{Q'} + \frac{R}{1 + b} (P')^{N-2} \right] \right. + P_{AT} q_i \left[ \frac{1 - Q_{AT}^{N-2}}{P} + \frac{R}{1 + b} Q_{AT}^{N-2} \right] \right\}$$

Concerning the reliability of the RB strategy, we are interested in the probability of entering the "Failure" state which is equal to

$$P_{FRB} = Q_{AT} \left[ 1 - (P')^{N-1} \right]$$

**PARALLEL N-VERSION PROGRAMMING MODEL**

A model for parallel NV strategy is shown in Figure 3, where the transition probabilities are defined by:

$$P = \text{Probability at least } m \text{ versions agree}$$

$$P' = \text{Probability at least } m \text{ good versions if versions agree}$$

while majority $m$ is defined as $m = \left\lceil \frac{N+1}{2} \right\rceil$ for $N$ odd or even

Adoption of analogous assumptions as for RB leads to the following notation:

$$\mu_s = \mu_s(1 + a + d)^{-1} = \mu_s(1 + c)^{-1}$$

where $a = \frac{2}{t_s}$, $d = \frac{2}{t_s}$ and $c = a + d$, while $t_s$ is the average version setup time, and $t_s$ is the average decision and best result choice time.

The models presented in the paper have been introduced in order to achieve comprehensive and quantitative evaluation of the performances of the software fault-tolerance strategies. The developed queueing models are used for both time efficiency and reliability evaluation of the two main fault-tolerance strategies: the Recovery Blocks scheme and the N-Version Programming. Some of the obtained results are presented in the paper showing that the models can be useful for system designers in deciding which of the strategies should be used depending on system parameters.

**REFERENCES**


