ON THE PERFORMANCE OF SOFTWARE FAULT-TOLERANCE STRATEGIES

Article · January 1980

CITATIONS
57

READS
46

3 authors, including:

Aksenti Grnarov
Saints Cyril and Methodius University of Skopje
53 PUBLICATIONS **413** CITATIONS
SEE PROFILE

Algirdas Avizienis
University of California, Los Angeles
175 PUBLICATIONS **15,664** CITATIONS
SEE PROFILE

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 approaches[1,2] (limited to detection of software failures), and fault-tolerant software approaches[3–5] which allow to recover after a software failure occured, were proposed. Figure 1 presents a general model for unified interpretation of the two main fault-tolerance strategies: Recovery Blocks scheme[4,6] and N-Version Programming[5,7]. The Recovery Blocks scheme (*RB*) consists of three major entities: a primary alternate (A_1), a list of supplementary alternates ($A_2, \cdots, A_{N-1}$) 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 $N \geqslant 2$ 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 inexact) 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 $P_1, P_2, \cdots, P_N$ 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[7]. "Continue" is continuation of the program when *AT* is successful (for *RB* only). No processing time is required for this state. "Update" is execution of the decision strategy in *NV* followed by updating of all versions with the "best" result. "Recovery" means selection and initialization of the next alternate or version for *RB* or *NVS* respectively. For *NVP* (as well as for *RB* and *NVS* after the last alternate or version has been executed), it means a transition to "Safe Down and Repair" state, where a higher level (global) recovery is initiated. "Next Segment" is the outcome of this state meanning that all actions that return the system to the successful execution of this segment are 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 ($\bar{T}$) in the system with *RB* or *NV* fault-tolerance strategy, the queueing theory[8] is used. The program segments are considered as "customers" and "service rates" of states are defined as follows: μ_s is the average service rate for processing a segment in a perfect system without fault-tolerance strategy; $\mu_R = \frac{\mu_s}{R}$ 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 *NVS* model[9] will not be presented in this paper.

In the following we will assume that $p_*^* = 1 - q_*^*$ and $P_*^* = 1 - Q_*^*$ where "*" stands for any symbol.

RECOVERY BLOCKS MODEL

A *RB* model is shown in Figure 2, where the following notation is used for $i = 1, 2, \ldots, N-1$:

p_i = Probability of no faults in alternate A_i

P_{AT} = Probability of success of *AT*

P_i = Probability of success of A_i if A_i executed.

Q'_i = Probability of failure in the same way of A_i and *AT*

The service rate of state (alternate) i is given by $\mu_i = \frac{1}{\bar{t}_i}$ where the average alternate i processing time $\bar{t}_i = \bar{t}^s + \bar{t}_i^a + \bar{t}^{AT}$ while $\bar{t}^s$ is the average segment execution time, $\bar{t}_i^a$ is the average alternate i setup time, and $\bar{t}^{AT}$ is the average *AT* execution time.
For computational convenience, we use

$$\mu_i = \mu_s(1 + a_i + k)^{-1} = \mu_s(1 + b_i)^{-1}$$

+ 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.

1604-8/80/0000-0251$00.75 © 1980 IEEE

where $\mu_s = \frac{1}{\bar{t}^s}$, $a_i = \frac{\bar{t}_i^a}{\bar{t}^s}$, $k = \frac{\bar{t}^{AT}}{\bar{t}^s}$ and $b_i = a_i + k$

We also use the notation: $Q_i = c''_i q_i$ where $c''_1 = 1$, and $c''_j \in [1, \frac{1}{q_j}]$ for $j = 2, 3, \cdots, N-1$; and also $Q'_i = c'_i q_i$ where $c'_i \in [0,1]$ is the conditional probability P(A_i and AT fail in the same way if A_i failed).

The following assumption can be made for the model:

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

b) The probabilities of no faults in the alternates are the same, i.e., $p_i = p_a$, $c''_j = c''$, and $c'_i = c'$.

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 $P_{AT} = c \cdot p_{AT}$ where the coverage $c = [(1-\alpha)k + \alpha]$ and $p_{AT} = (1 - k \cdot q_0)$ while α is the minimal coverage and q_0 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[8], the average segment processing time in the system is given by:

$$\bar{T}_{RB} = \frac{1}{\mu_a}\left\{1 + Q_{AT}\,P'\left[\frac{1-(P')^{N-2}}{Q'} + \frac{R}{1+b}(P')^{N-2}\right] + P_{AT}\,q_a\left[\frac{1-Q^{N-2}}{P} + \frac{R}{1+b}Q^{N-2}\right]\right\}$$

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 = Probability at least m versions agree

P' = Probability at least m 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:

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

where $a = \frac{\bar{t}^a}{\bar{t}^s}$, $d = \frac{\bar{t}^d}{\bar{t}^s}$, and $v = a + d$; while $\bar{t}^a$ is the average version setup time, and $\bar{t}^d$ is the average decision and best result choice time.

For $i = 1, 2, \cdots, N$, we consider also: $p_i = p_v$; $q'_i = q' = c'' q_v$, where q'_i is the conditional probability P(V_i fails if any other version failed) while $c'' \in [1, \frac{1}{q_v}]$; and the factor $c'_i = c'$ to denote the conditional probability P(V_i fails in the same way as another failed version if V_i failed).

It follows that the average segment processing time in a system with NVP strategy is given by

$$\bar{T}_{NVP} = \frac{1}{\mu_v}\left[1 + \frac{R}{1+v}\left(1-P\right)\right], \text{ where}$$

$$P = Z + p_v^m + q_v\left\{\sum_{i=1}^{m-1} p_v^{m-i}\left[\sum_{j=1}^{L}\binom{L+i}{j}(q')^j(p')^{L+i-j}\right] + \sum_{j=0}^{L}\binom{N-1}{j}(q')^j(p')^{N-j-1}\right\}, \text{ and}$$

$$Z = q_v\left\{\sum_{j=0}^{L+1}\binom{N-1}{j}(p'')^j(q'')^{N-j-1} + \sum_{i=1}^{L+1} p_v^i\left[\sum_{j=0}^{L-i+1}\binom{N-i-1}{j}(p'')^j(q'')^{N-i-j-1}\right]\right\}$$

while $L = N-m-1$ $(L \geqslant 0)$, $q'' = c'q'$ and $p'' = 1 - c'q'$.
The probability of failure is given by

$$P_{FNVP} = Z$$

COMPARISON OF RB AND NV STRATEGIES

In this section, we present some of the results derived from the discussed models, as well as results for sequential NV with two-out-of-N decision strategy ($2NVS$). As measures for evaluation, we use: a) the relative increase of the average segment processing time in a system defined as $\delta_\beta = \frac{\bar{T}_\beta - \bar{t}^s}{\bar{t}^s}$, b) the probability of failure $P_{F\beta}$, where $\beta \in \{RB, 2NVS \text{ and } NVP\}$

In order to perform a coherent comparison, we define N_d, the number of diversified entities required by each strategy, as: RB, $N-1$ alternates and the AT; NV, N versions.

Figure 4 compares the relative time penalties introduced by use of RB, $2NVS$ and NVP strategies as functions of the diversified entities N_d, with the repair ratio R as a parameter. RB appears to be more sensitive to the value of R than NV. Furthermore for both $2NVS$ and NVP, the time penalty is considerably reduced when N_d grows; while for RB, the influence of N_d strongly depends on the reliability of the AT, as it can be seen for different values of AT processing time ratio k

Figure 5 presents the variations of the probabilities of failure for the considered strategies as functions of N_d; we consider also the influence of: RB, the reliability of the AT as function of the initial coverage α and the processing time ratio k; NV, the "correlation" factor c'. These results point out the criticallity of the AT on the reliability of RB and shows that the impact of c' with respect to NV is approximately constant in the case of $2NVS$ while it increases with the value of N_d in the case of NVP.

In both Figures, the values of other system parameters are assumed to be: $p_a = p_v = .999$, $q_0 = .001$, $c'' = 10$, $a = .1$, and $d = .1$.

CONCLUDING REMARKS

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

1. Yau, S. S. and Cheung, R. C., "Design of Self-Checking Software", *Proc. 1975 Int. Conf. Reliable Software,* pp. 450-457.

2. Ayache, J. M , et. al., "Observer: A Concept for On-Line Detection of Control Errors in Concurrent Systems", *Proc. 1979 Int. Symp. Fault-Tolerant Computing,* Madison, Wisconsin, June 1979, pp. 79-86.

3 Fischler, M. A., et al., "Distinct Software: An Approach to Reliable Computing", *Proc. 2nd USA-Japan Computer Conf.,* Tokyo, Japan, 1975, pp. 1-7.

4. Randell, B , "System Structure for Software Fault-tolerance", *IEEE Trans. Software Eng.,* June 1975, pp. 220-232.

5. Avizienis, A., "Fault-Tolerance and Fault-Intolerance: Complementary approaches to Reliable Computing", *Proc. 1975 Int. Conf. Reliable Software,* pp. 458-464.

6. Lee, P. A., et. al., "A Recovery Cache for the PDP-11", *Proc. 1979 Int. Symp. Fault-Tolerant Computing,* Madison, Wisconsin, June 1979, pp. 3-8.

7. Avizienis, A. and Chen, L., "On the Implementation of N-Version Programming for Software Fault-Tolerance During Execution", *Proc. of COMPSAC 77,* November 1977, pp. 149-155.

8. Kleinrock, L., *Queueing Systems, Vol. 1: Theory,* J. Wiley and Sons, 1975.

9. Grnarov, A , Arlat, J. and Avizienis, A., "Modeling of Software Fault-Tolerance Strategies", *Proc. 1980 Pittsburgh Modeling and Simulation Conf.,* Pittsburgh, Pennsylvania, May 1980.

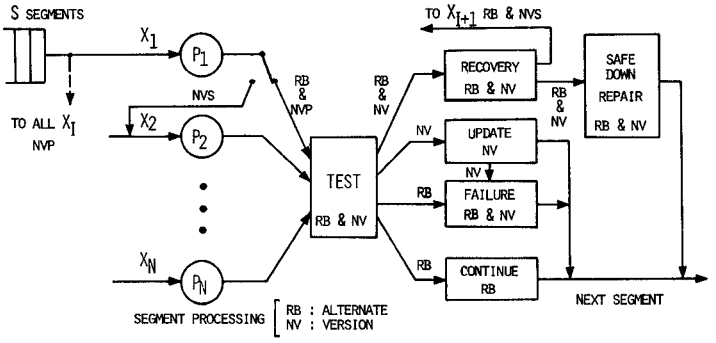

Figure 1 : General Model.

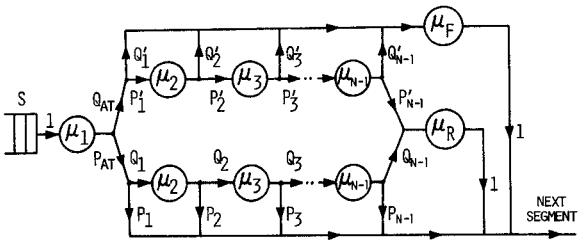

Figure 2 : A *RB* Model

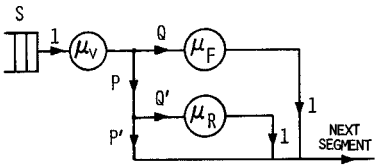

Figure 3 : A *NVP* Model.

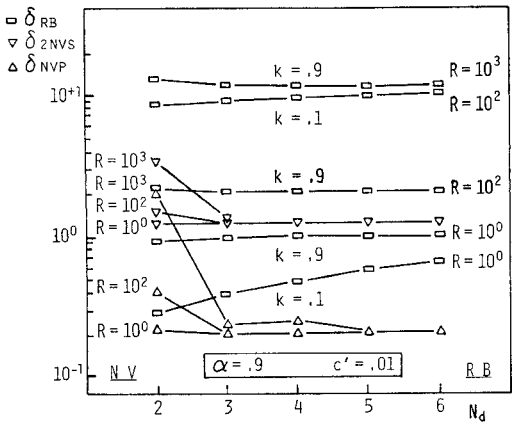

Figure 4 : Relative Increase of Processing Time.

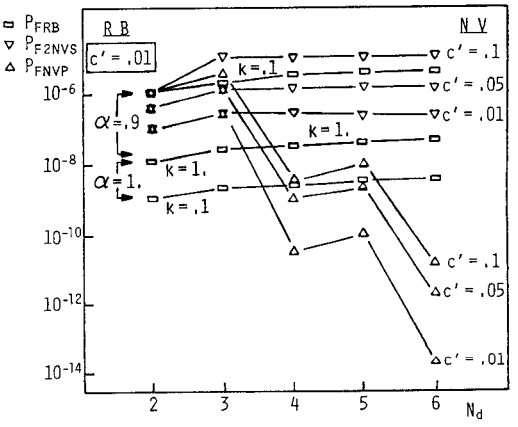

Figure 5 · Probability of Failure.