Classification and Unified Modeling for Duplication–based Recovery

Peter Sobe[1], Bernhard Fechner[2] and Jörg Keller[2]

[1] Universität Lübeck, Institut für Technische Informatik, Lübeck, Germany,
sobe@iti.uni-luebeck.de,
[2] FernUniversität Hagen, LG Parallelität und VLSI, Hagen, Germany,
{Bernhard.Fechner,Joerg.Keller}@fernuni-hagen.de

1 Introduction

For fault detection often process/thread duplication is used. A fault is detected by comparing the process states and outputs after a processing round. When detected, several ways to handle that situation exist, depending on the allowed redundancy in space and time (e.g. [1], [2]). Based on our recent work, we found a clear and comprehensive classification scheme for duplication–based recovery. Furthermore, the way of modeling is inspired by this classification and treats different recovery schemes in a unified way. The model contains parameters that reflect a common hardware or software basis used for process duplication, e.g. a common operating system or a common processor with two simultaneous instruction streams [3].

2 Classification of Recovery Schemes

A common way is to rollback processes to a saved state on stable storage and to retry. In one class, recovery consists solely of a retry in a single or both processes. That principle tries to reach a majority of fault-free states by state recalculation. Faults during that recalculation are covered. When both processes are used for that recalculation, we call that a **strongly pessimistic** scheme. In other classes, only a single process is used for retry and the other one is used for forward processing. If forward processing is done in a redundant fashion, e.g. time redundant by executing each round twice for fault detection this is called a **pessimistic scheme**. When the forward process does not employ redundancy, it is not able to detect additional faults, but reaches the best advance in application progress. Such a principle is called an **optimistic scheme**. Except for the strongly pessimistic scheme, the choice if a single guessed state (out of the two states from duplication) is used for forward processing or both states – with a half of resources (time or processing capacity) for each state – is still a design issue. Thus, we further distinguish **probabilistic** and **deterministic schemes**. For example, we classify the DMR-F forward recovery scheme [2] as a pessimistic deterministic one.

3 Reliability Modeling

A single round may be executed in several ways: (ff) fault-free, (1f) single-process-faulty, (2f) two-process-faulty when both processes exhibit a distinguishable malfunction, (cf) common-faulty, i.e. non-distinguishable faulty and (df) disastrous faulty, i.e. when the platform of duplication is crashed or destroyed in a way that recovery cannot be started anymore. At any time, the state of the process pair thus is $S \in \{\text{ff}, \text{1f}, \text{2f}, \text{cf}, \text{df}\}$. The probabilities of these states at the end of a round $P(S_{end}, S_{init})$, starting from any initial state S_{init} are taken from a Markov chain model. A multiple round analysis can be done by a combinatorial approach. All possible situations are described by a conjunction of single round results with their probabilities. Each fault scenario is determined by the type of the fault (out of S), the round number (i) when the fault is detected, and the time of saving the recent stable state for recovery. The distance from the last saved state on stable storage is indirectly available with the round number i, when states are regularly saved after every s rounds. Based on these situation-specific values, each recovery scheme displays a particular duration of recovery T_{recov}, probabilities for several cases of successful recovery P_{recov} and a number of rounds that are executed during recovery or that are lost during recovery N_{recov}. In order to describe fault-free and successfully tolerated fault scenarios, only $P(\text{ff}, \text{ff})$, $P(\text{1f}, \text{ff})$ and $P(\text{2f}, \text{ff})$ need to be calculated, shortly written as P_{ff}, P_{1f} and P_{2f}. Then, we can derive (a) the probability of i successive fault-free rounds, $C_{\mathit{ff}}(i)$=$(P_{\mathit{ff}})^i$, (b) the probability of state 1f after i rounds $C_{\mathit{1f}}(i)$=$P_{\mathit{ff}}^{(i\text{-}1)} P_{\mathit{1f}}$ and (c) the probability of state 2f after i rounds $C_{\mathit{2f}}(i)$=$P_{\mathit{ff}}^{(i\text{-}1)} P_{\mathit{2f}}$. Specifying the needed number of rounds n_r, P_{total} as the scheme reliability is calculated. The algorithm employs recursion and starts with round r=1:

If $r < n_r + 1$:

$$\begin{aligned} P_{total}(r) = {} & C_{\mathit{ff}}((n_r\text{+}1\text{-}r)) + \\ & \sum_{i=r}^{n_r} \sum_{z=1}^{n_{s,\mathit{1f}}} C_{\mathit{1f}}(i\text{-}r\text{+}1)\; P_{recov\ 1,z}(i)\; P_{total}(i\text{+}1\text{+}N_{recov\ 1,z}(i)) + \\ & \sum_{i=r}^{n_r} \sum_{z=1}^{n_{s,\mathit{2f}}} C_{\mathit{2f}}(i\text{-}r\text{+}1)\; P_{recov\ 2,z}(i)\; P_{total}(i\text{+}1\text{+}N_{recov\ 2,z}(i)) \end{aligned}$$

If $r = n_r + 1$: $\quad P_{total}(r) = 1$

P_{recov} and N_{recov} have two indices each, a first index specifies if the recovery starts from 1f or 2f, the second denotes a particular fault scenario during recovery. Their numbers $n_{s,\mathit{1f}}$ and $n_{s,\mathit{2f}}$ are known and for each scenario P_{recov} and N_{recov} can be specified. To guarantee termination, the recursion is limited by a maximum number of tolerated faults. Using that modeling approach – single rounds by Markov analysis and multiple rounds by a combinatorial model – we got coherent results, showing the tradeoff between recovery speed and reliability.

References

1. Grans, K.: DUSBER - a New Fault Tolerance Technique Combining Duplication and Recovery. Proceedings of EWDC-9, 110-113 (1998)
2. Long, J., Fuchs, W.K., Abraham, J.A.: A Forward Error Recovery Using Checkpoints in Distributed Systems. Int. Conf. on Parallel Processing, 272-275, IEEE Computer Society (1990)
3. Fechner, B., Keller, J., Sobe, P.: Performance Estimation of Virtual Duplex Systems in Simultaneous Multithreaded Processors. IPDPS 2004 Proceedings, IEEE Computer Society (2004)