Fast Abstracts

Twenty-Ninth Annual International Symposium on Fault Tolerant Computing

June 15-18, 1999
Madison, Wisconsin, USA

Sponsored by
IEEE Computer Society
Technical Committee on Fault-Tolerant Computing

In Cooperation with
IFIP WG 10.4 on Dependable Computing and Fault Tolerance

Supported by
The Department of Electrical and Computer Engineering of the University of Wisconsin-Madison
The Department of Electrical and Computer Engineering of the University of Iowa
The University of Illinois at Urbana-Champaign
AlliedSignal Inc.
The Madison Section of IEEE
Tandem (Compaq Computer Corp.)
International Business Machines

Scheduling Solutions for a Unified Approach to the Tolerance of Value and Timing Faults

F. Di Giandomenico and F. Grandoni
IEI/CNR
{digiandomenico,grandoni}@iei.pi.cnr.it

A. Bondavalli and I. Mura
CNUCE/CNR
{a.bondavalli,ivan.mura}@cnuce.cnr.it

Many critical applications require both correctness and timeliness. Proper solutions for these systems, called *responsive systems* [1], ask for approaches that integrate the fault tolerance and real-time facets. However, the two aspects have evolved for long time almost independently from each other. In real time contexts, the usual approach to cope with timing violations consists in preventing their occurrence. For each task, a worst case execution time (WCET) is determined, which is used by the scheduling policy to assign (statically or dynamically) the adequate amount of computing resources for the task to complete within its deadline. With this form of prevention, the compliance with timeliness requirements is guaranteed as long as the assumptions about the WCET of the tasks are verified. An accurate estimate of the task WCET is, however, a difficult issue. Moreover, even when such an estimate is known, it can result in a excessively pessimistic value, which is only encountered by the task in rare executions. Accepting such a pessimistic estimate for the WCET may result in an over-dimensioning of the time to be given to the task execution in most of the cases. Tolerance to violations of such assumptions (seen as faults in the time domain) would pave the way to much flexible and efficient system structuring.

Actually, existing fault tolerance solutions applied to real time contexts are mainly restricted to the tolerance to faults in the value domain. Solutions for integrating tolerance to both timing and value faults are being currently approached. An attractive direction of pursuing tolerance to timing faults is to identify a minimal implementation of the task, whose correct execution satisfies some minimum requirements, and building upon it a fault-tolerant scheme. This "critical" part, which has to be kept as much as possible reliable and verifiable, would be called in emergency situations to provide a minimum, though degraded, acceptable level of service.

The TAFT (Time Aware Fault-Tolerant) scheduling strategy [2] has been recently developed to provide a method for flexible and predictable execution of tasks with hard timing constraints (no deadline violations allowed) in presence of timing faults. It also provides a flexible implementation base to enable an easy mapping of a variety of strategies for the tolerance of faults in the value domain [3], thus configuring as a nice solution to the integrated tolerance of both timing and value faults .

A TAFT component is structured in a task pair (TP), made of a *nominal* part and of an *exceptional* part. The nominal part represents the normal computation of the task, and the exceptional one is a coarse implementation of it, in charge of performing a minimum acceptable amount of actions to consider the TP as correctly executed (although in a degraded form). Because of its simplicity (compared with the nominal part), the WCET of the exceptional part can be assumed to be known with a satisfactory level of accuracy. The nominal part is tried first and, if it does not complete successfully (because of either a timing or a value fault), the exceptional part is then run. On the contrary, if the nominal part succeeds, the exceptional part is not executed.

It is worthwhile remarking the advantages offered by this structuring of the TP. The critical part of the task that has to be necessarily executed to provide guarantees about the correctness (both in the time and values domains) of the TP, is precisely identified and confined inside the exceptional part. In most real-time systems, critical tasks are usually dealt with by resorting to a guarantee strategy, which provides assurance of timely execution. This same mechanism is adopted for the exceptional part of the TP: a time-window for the execution of the exceptional part is assigned, and this guarantees that at least the minimal amount of actions required to safely complete the task are performed. The nominal part of the TP, instead, being freed of the safety issues can be managed in a more flexible fashion. For instance, dynamic schedulers such as EDF can be employed to maximise processor utilisation, or best-effort based scheduling mechanisms can be applied towards system optimisation. As a beneficial side effect, a clever scheduling that completes many nominal parts will be able to get back the time that had been reserved for the execution of the corresponding exceptional parts. The scheduling of the TP nominal parts is thus a crucial activity.

It is easily understandable that flexible dynamic strategies would to be preferred in order to gain in system performability. Actually, although the execution of the exceptional part is sufficient to guarantee the safe execution of the entire TP, the correct execution of the nominal part brings higher benefits to the system and is, hence, highly desirable.

Our on-going work has the aim of comparing various scheduling algorithms for the nominal parts of the TP. The final objective is to find guidelines to the choice of the most appropriate scheduling strategy for the TP in relation with the application requirements and the offered workload. For instance, a scheduling algorithm that yields an high number of nominal parts to be missed points out that the system spends a significant period of its life in a degraded mode. While this would be acceptable if safety is a primary concern for the considered application, it would be inadequate in other cases. For example, when performance or reliability requirements are stringent, it would be preferred to adopt policies allowing only occasional exceptional behavior (i.e., nominal parts are mostly executed), even at the price of sacrifying the execution of some entire TP.

The impact of using specific dynamic scheduling strategies to accommodate the execution of the nominal parts is being evaluated through a simulation approach . A Monte-Carlo simulator has been set up, which generates workloads composed of periodic and sporadic tasks, characterised by different length and criticality, this last expressed in terms of quantified values associated to the correct/incorrect execution of the nominal part [4] (respectively, gain and penalty) and to the missed execution of the entire TP (a loss, only for aperiodic TP, whose exceptional parts cannot be pre-allocated). A number of dynamic scheduling strategies [5] for the nominal parts have been selected and implemented in the simulator, namely EDF, Least Laxity First, value based scheduling algorithms (value density, maximum gain, minimum loss, etc..). Execution times for the nominal parts are generated from a number of distributions, including infinite support ones (e.g., exponential and normal) to exploit the usage of the TP in a variety of timing violations situations. Temporal firewalls are introduced to make up for inaccurate estimations of the task computation times. Firewalls stop the execution of the nominal part should it last more than a predefined duration (determined on the basis of some statistical estimate of execution time duration) to stop excessively long executions in favour of an higher number of shorter ones. In order to model the behavior of more "optimistic" strategies (compared to those based on the WCET), execution times, by which a sufficient highly percentage of tasks complete, are also used as stopping time for nominal task executions.

Simulation experiments are currently in progress, and the first results are already in line with showing that it makes significant difference for the system effectiveness which scheduling policy is adopted. Amongst the preliminary simulation studies, we have evaluated the performability [6] of the system, by exploiting the reward structure defined in terms of gains, penalties, and losses. The performability measure has been computed as the total cumulated reward. We have studied the effect that the workload intensity and the criticality levels of the application have on the performability of the system for the various scheduling algorithms.

References

[1] M. Malek, "Omniscience, consensus, autonomy: three tempting roads to responsiveness," in Proc. IEEE 14-th Symposium on Reliable Distributed Systems, Bad Neuenahr, Germany, 1995, pp. xii-xiv.

[2] M. Gergeleit, M. Mock and E. Nett, " T-CORBA: making object-oriented systems time-aware," Computer Systems Science & Engineering, Vol. 13, pp. 151-160, 1998.

[3] E. Nett, H. Streich, P. Bizzarri, A. Bondavalli and F. Tarini, "Adaptive software fault tolerance policies with dynamic real-time guarantees," in Proc. IEEE Second Workshop on Object Oriented Real-Time Dependable Systems WORDS'96, Laguna Beach, CA, 1996, pp.

[4] A. Bondavalli, F. Di Giandomenico and I. Mura, "An optimal value-based admission policy and its reflective use in real-time systems," International Journal of Time-Critical Computing Systems, Vol. 16, pp. 5-30, 1999.

[5] A. Burns and N. Audsley, "Scheduling Hard Real-Time Systems: a Review," Software Engineering Journal, Vol. 6, pp. 116-128, 1991.

[6] J. F. Meyer, "On Evaluating the Performability of Degradable Computing Systems," IEEE Transactions on Computers, Vol. C-29, pp. 720-731, 1980.