Available online at www.sciencedirect.com
SCIENCE DIRECT®
J. Parallel Distrib. Comput. 64 (2004) 629–648
Journal of Parallel and Distributed Computing
http://www.elsevier.com/locate/jpdc

Efficient overloading techniques for primary-backup scheduling in real-time systems

R. Al-Omari,[a,1] Arun K. Somani,[b] and G. Manimaran[b,*]

[a] *Eclipz Nest Pervasive, System Group, IBM Austin, TX, USA*
[b] *Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011, USA*

Received 11 December 2002; revised 22 March 2004

Abstract

In real-time systems, tasks have deadlines to be met despite the presence of faults. Primary-Backup (PB) scheme is one of the most important schemes that has been employed for fault-tolerant scheduling of real-time tasks, wherein each task has two versions and the versions are scheduled on two different processors with time exclusion. There have been techniques proposed for improving schedulability of the PB-based scheduling, of which Backup–Backup (BB) overloading is among the most popular ones. In this technique two or more backups can share/overlap in time on a processor. In this paper, we propose two new techniques that accommodate more tasks and/or tolerate faults effectively. In the first technique, called dynamic grouping, the processors are dynamically grouped into logical groups in order to achieve efficient overloading of tasks on resources, thereby improving the schedulability and the reliability of the system. In the second technique, called PB overloading, the primary of a task can share/overlap in time with the backup of another task on a processor. The intuition is that, for a primary (or backup), the PB-overloading can assign an earlier start time than that of the BB-overloading, thereby increasing the schedulability. We conduct schedulability and reliability analysis of the proposed techniques through simulation and analytical studies. Our studies show that dynamic grouping improves the schedulability more than static grouping, and offers graceful degradation with increasing faults. Also, PB-overloading improves the schedulability more than BB-overloading, and offers reliability comparable to that of BB-overloading. The proposed techniques are generic that they can be incorporated into many fault-tolerant non-preemptive scheduling algorithms.

Keywords: Real-time systems; Multiprocessors; Scheduling; Fault-tolerance; Schedulability; Reliability

1. Introduction

In real-time systems the correctness of the results depend not only on the logical correctness, but also on the time at which the results are produced [20]. Multiprocessors and multicomputers based systems have emerged as a powerful computing means for real-time applications such as avionic control and nuclear plant control, because of their capability for high performance and reliability. The problem of scheduling real-time tasks in such systems is to determine when and on which processor a given task is executed. The allocation and scheduling can be performed either statically or dynamically. In static algorithms [19], the assignment of tasks to processors and the time at which the tasks start execution are determined a priori. Static algorithms are often used to schedule periodic tasks and are not applicable to aperiodic tasks whose arrival times and deadlines are not known a priori. Scheduling such tasks requires a dynamic scheduling algorithm [14,21].

In dynamic scheduling, when a new set of tasks arrive at the system, the scheduler determines the feasibility of scheduling these new tasks without jeopardizing the guarantees that have been provided for the previously scheduled tasks. For predictable executions, schedulability analysis must be done before tasks' execution begun. For feasibility analysis, the tasks' worst case computation times must be taken into account. A feasible schedule is generated if timing and other

*Corresponding author.
E-mail addresses: alomari@us.ibm.com (R. Al-Omari), arun@iastate.edu (A.K. Somani), gmani@iastate.edu (G. Manimaran).
[1] This work was done when the author was at Iowa State University.

0743-7315/$ - see front matter © 2004 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2004.03.015

requirements of the tasks can be satisfied, i.e., if the schedulability analysis is successful. Tasks are dispatched according to this feasible schedule.

Due to the critical nature of tasks in a hard real-time system, it is essential that every task admitted in the system completes its execution even in the presence of faults. Therefore, fault-tolerance is an important requirement in such systems [5,8]. Scheduling multiple versions of tasks on different processors provides fault-tolerance [6,7,9–13,15,17,18,23,25,26,28]. Overload handling techniques to deal with timing faults are proposed in [16,22]. One of the approaches that is used for fault-tolerant scheduling of real-time tasks is the primary-backup (PB) model, in which two versions of a task are scheduled on two different processors and an *acceptance test* is used to check the correctness of the execution result [7,10,15,23]. The backup version is executed only if the output of the primary version fails the acceptance test, otherwise it is deallocated from the schedule. In [12], processor failures are handled by maintaining backup or contingency schedules. The backup schedules are generated assuming that a (optimal) schedule exists and is enhanced with the addition of "ghost" tasks which function as backup tasks.

PB-based fault-tolerant scheduling has been enhanced through various techniques such as adaptive scheduling and overloading techniques. In adaptive fault-tolerant scheduling, there are two types of adaptiveness reported in the literature: (1) scheduling algorithms that allow primary and backup versions to overlap in execution and control the overlap interval in a continuous manner based on the fault rate observed in the system [3]; (2) scheduling algorithms that exploit the flexibility provided by the tasks in terms of selecting the suitable fault-tolerant scheme, at run-time, based on the current state of the system. The adaptive scheduler reported in [8] belongs to the latter category, which selects one of Triple Modular Redundancy (TMR), PB, or Primary-Exception (PE) schemes, at run-time, based on the system state.

In the context of fault-tolerant scheduling, the term "overloading" refers to scheduling of more than one task (version) on the same/overlapping time interval on a processor. Fault-tolerant scheduling algorithms [6,7,15] have employed overloading techniques as a means to conserve system resources, thereby improving the schedulability of the system.

1.1. Backup overloading

Backup overloading [6,7] is defined as scheduling backups of multiple primaries onto the same or overlapping time interval on a processor. Fig. 1 shows two primaries (Pr_1 and Pr_2) that are scheduled on processors 1 and 3, respectively, and their backups (Bk_1 and Bk_2) are scheduled in an overloading manner on processor 2. The following are the conditions under which backups can be overloaded on a processor:

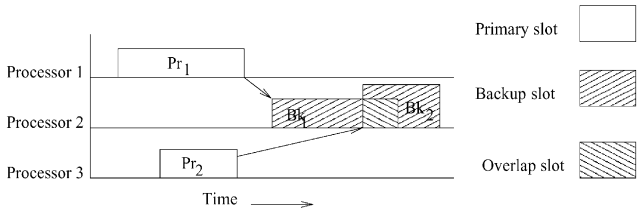

Fig. 1. Backup overloading.

Condition 1: The primaries of the backups being overlapped must be scheduled onto different processors.

Condition 2: At most one of these primaries is expected to encounter a fault.

Condition 3: At most one version of a task is expected to encounter a fault. In other words, if the primary of a task fails, its backup is expected to succeed.

Condition 1 is needed to handle permanent faults. Condition 2 is needed to ensure that at most one backup is required to be executed among the overloaded backups. Condition 3 is needed to ensure that at least one version of each task is executed without any fault. Condition 2 of backup overloading can be guaranteed by Assumption 1 (stated in Section 2.4). Condition 3 is guaranteed by Assumption 2 (stated in Section 2.4) which states that the *MTTF* (mean time to failure) of the system is much greater than *TTSF* (time to second failure) for the fault-tolerant techniques.

1.2. Previous work and motivation for our work

In this section, we first discuss existing overloading and grouping techniques and then discuss the motivation for dynamic grouping and PB-overloading.

BB-overloading: *No-grouping* [6,7]: A scheduling algorithm that uses Backup–Backup (BB) overloading was proposed in [6,7] for fault-tolerant dynamic scheduling of real-time tasks in multiprocessor systems. In this algorithm, a backup has $n-1$ choices for overloading with another backup (except the processor onto which its primary is scheduled), where n is the number of processors in the system. Thus, the number of backups that could potentially be overloaded (in a time slot) is $n-1$. Although this algorithm has a potential to offer higher schedulability due to its maximum flexibility in overloading, it can tolerate at most one failure at any point of time. Which is too optimistic and becomes unrealistic for larger n as *MTTF* becomes small.

BB-overloading: *Static grouping* [15]: This algorithm statically divides the system processors into disjoint logical groups and allows backup overloading to take place only within the group. That is, the processors onto