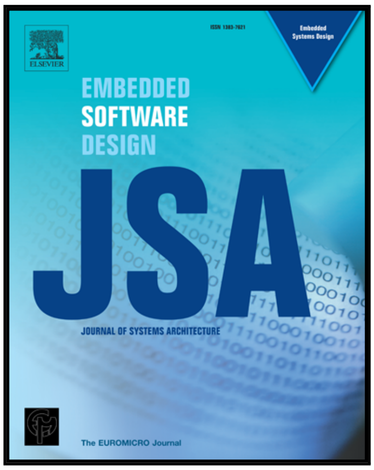

Task Scheduling with Fault-Tolerance in Real-Time Heterogeneous Systems

Jing Liu, Mengxue Wei, Wei Hu, Xin Xu, Aijia Ouyang

PII: S1383-7621(18)30175-9
DOI: https://doi.org/10.1016/j.sysarc.2018.08.007
Reference: SYSARC 1521

To appear in: *Journal of Systems Architecture*

Received date: 23 April 2018
Revised date: 18 July 2018
Accepted date: 26 August 2018

Task Scheduling with Fault-Tolerance in Real-Time Heterogeneous Systems

Jing Liu[1,2], Mengxue Wei[1,2], Wei Hu[1,2], Xin Xu[1,2], and Aijia Ouyang[3]

[1]College of Computer Science and Technology, Wuhan University of Science and Technology, China
[2]Hubei Province Key Laboratory of Intelligent Information Processing and Real-time Industrial System
[3]School of Information Engineering, Zunyi Normal University
[*]Corresponding author E-mail: Idealer@126.com

***Abstract*—Nowadays, the performance of heterogeneous systems has been improved dramatically, which also increases the complexity of heterogeneous systems, leading to the growing potential of system failures. Failures can be masked through scheduling approaches. Efficient task scheduling with fault-tolerance can guarantee the execution of tasks and satisfy the real-time nature. In this paper, we address the problem of scheduling tasks on heterogeneous systems with the target to support the maximum number of permanent failures while meeting a given time constraint. The problem is NP-hard and we propose a heuristic algorithm DBSA to solve it. DBSA can dynamically calculate the number of tolerating permanent failures. Firstly, the makespan when systems tolerate a fixed number of failures is calculated. Then, DBSA gets the actual number of tolerating failures by constantly comparing the makespan with a given deadline. Finally, DBSA maps tasks to appropriate processors without violating precedence constraints. Experimental results demonstrate that DBSA can effectively tolerate failures and improves system reliability.**

***Index Terms*—Fault-tolerance, permanent failures, scheduling algorithms, reliability, time constraint.**

I. Introduction

MANY heterogeneous systems, especially safety-critical systems, have real-time and fault-tolerance requirements, like automotive, aircraft and space shuttle [1]. That means such systems must guarantee their functional requirements with time constraints even in the presence of faults. Scheduling plays a substantial role in the performance of real-time systems. Effective fault-tolerant scheduling will greatly improve the reliability of systems while deadlines are satisfied. Therefore, there is an urgent need for researchers to study effective fault-tolerant scheduling algorithms to handle faults within time constraints.

Due to the effects of hardware defects, electromagnetic interferences and cosmic ray radiations, failures may occur at run-time [2]. Because the occurrence of failures is often unpredictable, fault-tolerance is considered in the design of real-time scheduling algorithms to improve the system reliability [3]. Real-time scheduling with fault-tolerant capability is implemented using redundancy either in time or space [4]. Usually, time redundancy is a viable and cost-efficient means to achieve fault-tolerance in many resource-constrained systems [1]. In this paper, we adopt time redundancy to support fault-tolerance while ensuring a time constraint. There are two main approaches to solve faults in time redundancy: the active scheme and the passive scheme. In these two schemes, each task has several copies which can be considered as a primary copy and multiple backup copies. In the active scheme, primary executes simultaneously with its backups, that is to say backups can begin to execute even though its primary does not fail [5]. If primary fails, backups will be responsible for critical functionality in the application [6]. On the contrast, the backups are allowed to execute only if the primary fails in the passive scheme [7]. This scheme assumes that real-time tasks must have enough laxity to re-execute their backups [8]. However, this assumption is unrealistic in practice, particularly when real-time systems are heavily loaded. The principal difference between these two approaches is that the active scheme entails forward error masking: all multiple copies of task are executed. In the passive scheme, a backup is executed only if the acceptance test flags a failure. The passive scheme assumes that a fault detection mechanism is employed to detect failures while the active scheme does not need this mechanism. Therefore, in this paper, we choose the active scheme to support fault-tolerance.

The two fault-tolerant approaches mentioned above are used to mask failures which can be transient or permanent [9]. When a transient failure occurs, the failed processor may soon recover and resume its duty. When a permanent failure occurs, the corresponding processor can not execute any tasks. In order to ensure the normal running of systems, tasks in the failed processor must be migrated to be executed on other available processors. It seems that comparing with transient failures, permanent failures are more intractable to deal with. In this paper, we focus on permanent failures. Using time redundancy to solve failures will inevitably increase the scheduling length of applications, which may make tasks miss deadline. Thus, how to efficiently schedule tasks with fault-tolerance while satisfying time constraint requirement has been identified as one of the most fundamental issues.

Many fault-tolerant scheduling algorithms have been developed in the last decade [10][11][12][13][14][15][16][17]. Zhao et al.[10] study energy efficient reliability scheduling targeting precedence constrained tasks, and each task has its own deadline. However, they do not consider communication time. Guo et al.[11][12] study similar problems and they focus on periodic tasks. Haque et al.[13] and Han et al.[14]