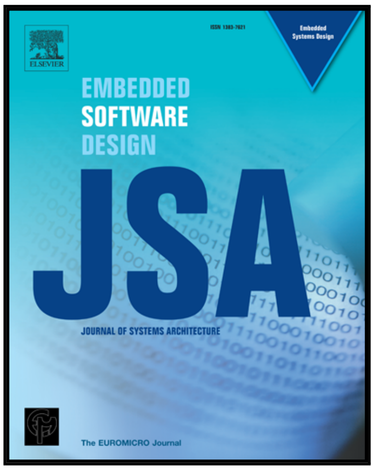

Bounding Carry-in Interference for Synchronous Parallel Tasks under Global Fixed-Priority Scheduling

Meiling Han, Tianyu Zhang, Qingxu Deng

PII: S1383-7621(18)30012-2
DOI: https://doi.org/10.1016/j.sysarc.2018.08.004
Reference: SYSARC 1518

To appear in: *Journal of Systems Architecture*

Received date: 14 January 2018
Revised date: 28 June 2018
Accepted date: 7 August 2018

Bounding Carry-in Interference for Synchronous Parallel Tasks under Global Fixed-Priority Scheduling

Meiling Han, Tianyu Zhang, Qingxu Deng*

Northeastern University, China

Abstract

With the increasing trend towards using multi-core architecture for embedded systems, the study of intra-task parallelism becomes attractive and desirable in the literature. Although several work studying parallel task models has been proposed, the problem of precise scheduling analysis for the multiprocessor case has largely remained open. To this end, this paper concentrates on analyzing the response time for synchronous parallel real-time tasks scheduled on a multiprocessor platform. Specifically, by exploring the feature of each interfering task, we first present an interference analysis method with higher accuracy compared to other existing work. Considering the cost brought by a high complexity of the proposed method, we further introduce techniques to increase the efficiency with an acceptable loss of accuracy which gives more flexibility to the system designers. Finally, we provide a dynamic programming algorithm for analyzing the schedulability of the whole task set based on our proposed interference analysis technique. Experimental evaluation validates the performance and efficiency of the proposed approach by comparing with other methods.

Keywords: Embedded real-time systems, global fixed-priority, multiprocessor systems, response time analysis, synchronous parallel tasks

1. Introduction

Due to the increasing demand of the functionality and service quality, embedded real-time systems are shifting from single-core to multi-core processors. Applications are also required to fully exploit the computation capacity of multiprocessor platforms. To achieve this, designers are committed to design parallelized applications. However, traditional researches on real-time scheduling assume that applications are executed sequentially, which means any particular task is allowed to be executed upon at most one processor at each time instant. Several techniques (e.g.,[10, 4, 13, 15, 18]) have been proposed to migrate the traditional schedulability analysis techniques to be used for parallelism settings. However, such a migration is non-trivial and several challenges need to be tackled.

Worst case response time analysis for parallel tasks becomes much more complex compared to that for sequentially executed tasks. Recently, there have been several promising techniques developed for parallel tasks schedulability analysis (e.g., [13, 3, 2, 10]) and response time analysis (e.g., [11, 17, 20]) under global multiprocessor scheduling. In this paper, we study response time analysis for *synchronous parallel tasks* (sp-tasks) under global fixed-priority (G-FP) scheduling. In the sp-task model, each task is composed of a certain number of segments each of which contains several threads. A thread of a segment cannot start executing until all threads of the previous segment have been completed.

The challenge of analyzing the worst case response time of sp-tasks under G-FP scheduling comes from how to upper-bound the total interference of higher priority tasks. Moreover, intra-task parallelism of sp-tasks further aggravates this problem and brings significantly high complexity to the analysis. To tackle this, [20] assumed that the total executions of a task are executing in parallel on all processors. Then, the worst case workload of a parallel task generated in an interval can be computed base on the worst case scenario assumptions for a sequential task. However, such an assumption in [20] may lead pessimistic analysis. Some studies (e.g., [17, 10, 3, 14]) computed the interference of higher priority tasks in a much more precise way and they assumed that all higher priority tasks have carry-in, which is defined as the first instant of a sp-task executed in the interval with release time before and finish time after the

*Corresponding author

Email address: dengqx@mail.neu.edu.cn (Qingxu Deng)