ELSEVIER

Journal of Systems Architecture 46 (2000) 931–949

JOURNAL OF SYSTEMS ARCHITECTURE

www.elsevier.com/locate/sysarc

Measurement based analysis of temporal behaviour as support for scheduling problems in parallel and distributed real-time systems [☆]

F.J. Suárez [*], D.F. García, J. García

Area de Arquitectura y Tecnología de Computadores, Departamento de Informática, Campus de Viesques s/n, Edificio Dept. (despacho 1.2.13), Universidad de Oviedo, 33204 Gijón, Spain

Received 30 June 1998; received in revised form 3 September 1999; accepted 17 September 1999

Abstract

Static analysis, based on scheduling techniques, provides the most typical approach for validation of real-time systems. However, in the case of complex real-time systems such as parallel and distributed systems, many simplifications are made in order to make analysis tractable. This means that even if the system can be statically validated, the real behaviour of the system in execution may be different enough from its theoretical behaviour to make it invalid. In these cases, an analysis based on measurement of the system in execution constitutes an invaluable aid to the static analysis. This article describes a methodology for the analysis of the temporal behaviour of parallel and distributed real-time systems with end-to-end constraints. The analysis is based on the measurement of a prototype of the system in execution and is supported by a behavioural model. The main components of the model are the sequences of activities throughout the system tasks (*transactions*), which are carried out in response to input events, causing the corresponding output events. Thus, the temporal behaviour of the system is viewed as a set of real-time transactions competing for the available resources. This article also includes experimental results of applying the methodology to the analysis of a well-known case study. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Real-time; Parallel and distributed systems; Measurement; Analysis methodology

1. Introduction

Timing analysis and validation have been highlighted as key areas of research for real-time and embedded computing [16]. Static analysis, based on formal methods and scheduling theory, and dynamic analysis, based on simulation and run-time measurement, are the two approaches most commonly used for timing analysis and validation.

In complex industrial real-time systems, commonly implemented as parallel or distributed systems, timing constraints must be derived and imposed based on the end-to-end requirements between the input events (usually from the external environment of the system), and the output events (generated by the system in response to the input events). This end-to-end approach has been

[☆] This research work has been supported by the ESPRIT HPC 8169 project ESCORT.

[*] Corresponding author. Fax: +34-985182228.

E-mail address: fran@atc.uniovi.es (F.J. Suárez).

followed by several authors in the real-time computing area, working with mono-processor systems [3], parallel systems [18] and distributed systems [6,13].

Real-time scheduling, the most common approach in the related literature, produces low degrees of success with systems whose *end-to-end tasks* may have complex synchronization patterns (i.e. not only pipeline patterns, but fork-join patterns as well) and widely varying releases, execution times and resource requirements. The problem with scheduling theory for this kind of systems comes from the fact that the models we use incorporate many simplifications in order to make the problem tractable. Thus, a system which is perfectly schedulable in theory, may not be schedulable in practice, after implemententation on a specific platform.

As real-time systems become more complex and more dynamic, analysis based on measurement provides more help in understanding their temporal behaviour during execution. Thus, this kind of analysis can effectively help in bridging the gap between real-time scheduling theory and implementation realities.

Here, we propose an analysis methodology of temporal behaviour of parallel and distributed real-time systems based on measurement, as a complement to the static analysis based on scheduling. The methodology starts with an initial design for the real-time system, which is the result of using a suitable design methodology for this kind of systems [7]. After building a system prototype and carrying out measurements during its execution, the methodology permits not only the real-time system to be checked in practice, but also the identification of a set of causes for the observed system behaviour. Knowledge of these causes can also help in refining the scheduling parameters (i.e. task priorities and mapping) when the timing constraints for the system are not fulfiled.

The methodology is supported by metrics corresponding to three system views [19], which are behavioural, structural and resource views. The structural view is a static view of the system and provides information about the software structure and its mapping on the hardware. The resource view provides dynamic information about the use of the resources. The last view, the behavioural view, provides information about the temporal behaviour of the system in terms of sequences of activities carried out in the system.

The organization of the rest of the article is as follows: Section 2 presents the computational model of the kind of real-time systems to which the methodology is oriented; in Section 3, the model used in the behavioural view is shown; Section 4 presents the main steps and aspects of the analysis methodology; in Section 5, a well-known case study is analysed using the methodology; finally, in Section 6, the conclusions and future work are presented. Although the methodology can be applied to real-time systems in general, this work deals with parallel and distributed real-time systems, hereafter simply referred to as *real-time systems* (RTS).

2. Computational model

The real-time applications supported by the methodology are composed of sequences of tasks, called *end-to-end tasks*. These tasks are executed in order to give response to input events in the system (internal clock signals or external signals from the transducers), generating the corresponding output events (signals for the actuators). A data or control dependency exists between the successive tasks, which imposes a precedence requirement on their scheduling. The sequences of tasks may span multiple processors and one simple task can belong to more than one *end-to-end task*.

Tasks are the units of scheduling in the computational model, but they are not the only components of the model. In fact, because we consider the possibility of a certain level of complexity in the tasks, we also consider the blocks of code executed within the tasks in the computational model. In the simplest case, only one block of code (hereafter called *block*) corresponds to the whole code of one task. The existence of computational units within the task in our model will also permit the construction of a more detailed behavioural model to support the analysis of *end-to-end tasks*, as will be shown later.

دریافت فوری ← متن کامل مقاله

ISIArticles
مرجع مقالات تخصصی ایران

✓ امکان دانلود نسخه تمام متن مقالات انگلیسی
✓ امکان دانلود نسخه ترجمه شده مقالات
✓ پذیرش سفارش ترجمه تخصصی
✓ امکان جستجو در آرشیو جامعی از صدها موضوع و هزاران مقاله
✓ امکان دانلود رایگان ۲ صفحه اول هر مقاله
✓ امکان پرداخت اینترنتی با کلیه کارت های عضو شتاب
✓ دانلود فوری مقاله پس از پرداخت آنلاین
✓ پشتیبانی کامل خرید با بهره مندی از سیستم هوشمند رهگیری سفارشات