Automatica 64 (2016) 44–53

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Design and implementation of distributed resource management for time-sensitive applications[☆]

Georgios C. Chasparis[a], Martina Maggio[b], Enrico Bini[b], Karl-Erik Årzén[b]

[a] Software Competence Center Hagenberg GmbH, Softwarepark 21, A-4232 Hagenberg, Austria

[b] Department of Automatic Control, Lund University, Sweden

ARTICLE INFO

Article history:
Received 4 June 2013
Received in revised form
15 April 2015
Accepted 21 August 2015
Available online 18 November 2015

Keywords:
Resource management
Distributed optimization
Real-time systems

ABSTRACT

In this paper, we address distributed convergence to *fair* allocations of CPU resources for time-sensitive applications. We propose a novel resource management framework where a centralized objective for fair allocations is decomposed into a pair of performance-driven recursive processes for updating: (a) the allocation of computing bandwidth to the applications (*resource adaptation*), executed by the resource manager, and (b) the service level of each application (*service-level adaptation*), executed by each application independently. We provide conditions under which the distributed recursive scheme exhibits convergence to solutions of the centralized objective (i.e., fair allocations). Contrary to prior work on centralized optimization schemes, the proposed framework exhibits adaptivity and robustness to changes both in the number and nature of applications, while it assumes minimum information available to both applications and the resource manager. We finally validate our framework with simulations using the TrueTime toolbox in MATLAB/Simulink.

© 2015 Elsevier Ltd. All rights reserved.

1. Introduction

The current trend in embedded computing demands that the number of applications sharing the same execution platform increases. This is due to the increased capacity of the new hardware platforms, e.g., through the use of multi-core techniques. An example includes the move from federated to integrated system architectures in the automotive industry (Di Natale & Sangiovanni-Vincentelli, 2010).

In such scenarios, the need for better mechanisms for controlling the rate of execution of each application becomes apparent. To this end, virtualization or resource reservation techniques (Abeni & Buttazzo, 1998; Mercer, Savage, & Tokuda, 1994) are used. According to these techniques, each reservation is viewed as a *virtual processor* (or *platform*) executing at a fraction of the speed of the physical processor, i.e., the *bandwidth* of the reservation. An orthogonal dimension along which the performance of an application can be tuned is the selection of its *service level*. It is assumed that an application is able to execute at different service levels, where a higher service level implies a higher quality-of-service (QoS). Examples include the adjustable video resolutions and the adjustable sampling rates of a controller.

Typically this problem is solved by using a *resource manager* (RM), which is in charge of: (a) *assigning virtual processors to the applications*, (b) *monitoring the use of resources*, and (c) *assigning the service level to each application*. The goal of the RM is to maximize the overall delivered QoS. This is often done through *centralized optimization* and the use of *feedback* from the applications.

RM's that are based on the concept of feedback, monitor the progress of the applications and adjust the virtual platforms based on measurements (Eker, Hagander, & Årzén, 2000; Steere et al., 1999). In these early approaches, however, quality adjustment was not considered. Instead, Ref. Cucinotta, Palopoli, Abeni, Faggioli, and Lipari (2010) proposed an inner loop to control the resource allocation nested within an outer loop that controls the overall delivered quality.

Optimization-based resource managers have also received considerable attention (Lee, Lehoczky, Siewiorek, Rajkumar, & Hansen, 1999; Rajkumar, Lee, Lehoczky, & Siewiorek, 1997). These

☆ The research leading to these results was supported by the Linnaeus Center LCCC, the Swedish VR project no. 2011-3635 "Feedback-based resource management for embedded multicore platform", and the Marie Curie Intra European Fellowship within the 7th European Community Framework Programme. The material in this paper was partially presented at the 2013 American Control Conference, June 17–19, 2013, Washington, DC, USA. This paper was recommended for publication in revised form by Associate Editor Richard D. Braatz under the direction of Editor Christos G. Cassandras. An earlier version of this paper appeared in Chasparis et al. (2013) and its implementation framework appeared in Maggio et al. (2013).

E-mail addresses: georgios.chasparis@scch.at (G.C. Chasparis), martina.maggio@control.lth.se (M. Maggio), bini@control.lth.se (E. Bini), karlerik@control.lth.se (K.-E. Årzén).

http://dx.doi.org/10.1016/j.automatica.2015.09.015
0005-1098/© 2015 Elsevier Ltd. All rights reserved.

approaches, however, rely on the solution of a centralized optimization that determines *both* the amount of assigned resources and the service levels of all applications (Bini et al., 2011; Rajkumar et al., 1997; Sojka et al., 2011). In the context of networking, Ref. Johansson, Adam, Johansson, and Stadler (2006) models the service provided by a set of servers to workloads belonging to different classes as a utility maximization problem. However, there is no notion of adjustment of the service level of the applications.

An example of a combined use of optimization and feedback was developed in the ACTORS project (Årzén Romero Segovia, Schorr, & Fohler, 2011; Bini et al., 2011). In that project, applications provide a table to the RM describing the required amount of CPU resources and the expected QoS achieved at each supported service level (Årzén et al., 2011; Bini et al., 2011). In the multi-core case, applications are partitioned over the cores and the amount of resources is given for each individual partition. Then, the RM decides the service level of all applications and how the partitions should be mapped to physical cores using a combination of Integer Linear Programming (ILP) and first-fit-decrease (FFD) for bin packing.

On-line centralized optimization schemes have several weaknesses. First, the complexity of the solvers used to implement the RM (such as ILP solvers) grows significantly with the number of applications. It is impractical to have a RM that optimally assigns resources at the price of a large consumption of resources by the RM itself. Second, to enable a meaningful formulation of a cost function in such optimization problems, the RM must compare the quality delivered by different applications. This comparison is unnatural because the concept of quality is extremely application dependent. Finally, a proper assignment of service levels requires application knowledge. In particular, applications must inform the RM about the available service levels and the expected consumed resources at each service level, increasing significantly communication complexity.

To this end, distributed optimization schemes have recently attracted considerable attention. Ref. Subrata, Zomaya, and Landfeldt (2008) considered a cooperative game formulation for job allocation to service providers in grid computing. Ref. Wei, Vasilakos, Zheng, and Xiong (2010) proposed a non-cooperative game formulation to allocate computational resources to a given number of tasks in cloud computing. Tasks have full knowledge of the available resources and try to maximize their own utility function. Similarly, in Grosu and Chronopoulos (2005) the load balancing problem is formulated as a non-cooperative game.

Contrary to the grid computing setup of Subrata et al. (2008) or the load balancing problem of Grosu and Chronopoulos (2005) and Wei et al. (2010), this paper addresses a lower-level resource allocation problem, that is, *the establishment of fair allocations of CPU bandwidth among time-sensitive applications which adjust their own service levels*. Contrary to the cloud computing setup of Wei et al. (2010), a game-theoretic formulation may not easily be motivated practically when addressing such lower-level (single node) resource allocation problems. Instead, we propose a distributed optimization scheme, according to which a centralized objective for fair allocations is decomposed into a pair of performance-driven recursive processes for updating: (a) the allocation of computing bandwidth to the applications (*resource adaptation*), executed by the RM, and (b) the service level of each application (*service-level adaptation*), executed by each application independently. We provide conditions under which the distributed recursive scheme exhibits convergence to fair allocations.

The proposed scheme introduces a design technique for allocating computing bandwidth to *time-sensitive applications*, i.e., applications whose performance is subject to strict time deadlines, such as multimedia and control applications. In particular, the proposed scheme: (a) exhibits linear complexity with the number of applications, (b) drops the assumption that the RM has knowledge of application details, and (c) exhibits adaptivity and robustness to the number and nature of applications. This paper extends the theoretical contributions of Chasparis, Maggio, Årzén, and Bini (2013) by addressing global convergence and asynchronous updates. Furthermore, Ref. Maggio, Bini, Chasparis, and Årzén (2013) presents the full implementation framework in Linux.

The paper is organized as follows. Section 2 provides the overall framework, while Section 3 presents the distributed scheme for resource allocation. Section 4 presents the convergence behavior for the synchronous and asynchronous case. Section 5 presents technical details required for the derivation of the main results in Section 4. Section 6 provides selective simulations. Finally, Section 7 presents concluding remarks.

Notation:

- $\Pi_{[a,b]}$ is the projection onto the set $[a,b]$.
- For some finite sequence $\{x_1, x_2, \ldots, x_n\}$ in $\mathbb{R}$, define $\mathrm{col}\{x_1, x_2, \ldots, x_n\}$ to be the column vector in $\mathbb{R}^n$ with entries $\{x_1, x_2, \ldots, x_n\}$.
- For any $x \in \mathbb{R}$, define the operator $[x]_-$ as follows:

$$[x]_- \triangleq \begin{cases} x, & x \le 0 \\ 0, & x > 0. \end{cases}$$

- For any $x \in \mathbb{R}^n$ and set $A \subset \mathbb{R}^n$, define $\mathrm{dist}(x, A) \doteq \inf_{y\in A} \|x-y\|$, where $\|\cdot\|$ denotes the Euclidean norm.
- For some finite set A, $|A|$ denotes the cardinality of A.

2. Framework and problem formulation

2.1. Resource manager and applications

The overall framework is illustrated in Fig. 1. A set $\mathcal{I}$ of n (time-sensitive) applications is sharing the same CPU platform. Let i be a representative element of this set. Since we allow applications to dynamically join or leave, the number n may not be constant over time.

The resources are managed by a RM that allocates resources through a Constant Bandwidth Server (CBS) (Abeni & Buttazzo, 1998) with period P_i and budget Q_i. Hence, application i is assigned a *virtual platform* with bandwidth $v_i = Q_i/P_i$ corresponding to a fraction of the computing power (or speed) of a single CPU. Obviously, not all virtual platforms v_i are feasible, since their sum cannot exceed the number κ of available CPU's. Formally, we define the set of ***feasible virtual platforms***, $(v_1, \ldots, v_n)$, as

$$\mathcal{V} \doteq \left\{ \mathbf{v} = (v_1, \ldots, v_n) \in [0,1]^n : \sum_{i=1}^{n} v_i \le \kappa \right\}. \quad (1)$$

In this study, the main concern is the computation of the allocation $\mathbf{v}$ in real time such that a centralized objective is achieved. However, we will not be concerned with the exact mapping of this allocation onto the available cores. Such mapping can be performed by a standard first-fit-decrease algorithm. Furthermore, in practice, more constraints might be present, especially if applications are single-threaded (i.e., they may only run on a single core). In this case, the above feasibility constraint will be a relaxed version of the original problem, however, the forthcoming analysis can be modified in a straightforward manner to incorporate additional constraints on $\mathcal{V}$.

Each application $i \in \mathcal{I}$ may change its *service level*, s_i. It represents a qualitative indicator of the delivered quality of application i, assuming sufficient amount of resources v_i. Naturally, it can be represented by a real number $s_i \in \mathcal{S}_i \doteq [\underline{s}_i, \infty) \subset \mathbb{R}$, where $\underline{s}_i > 0$ is the minimum possible service level of application i. The domain $\mathcal{S}_i$

Download English Version:

https://daneshyari.com/en/article/695308

Download Persian Version:

https://daneshyari.com/article/695308

Daneshyari.com