Available online at www.sciencedirect.com
ScienceDirect
The Journal of Systems and Software 80 (2007) 997–1004
The Journal of Systems and Software
www.elsevier.com/locate/jss

Feedback control-based dynamic resource management in distributed real-time systems

Tian He [*,1], John A. Stankovic, Michael Marley, Chenyang Lu [2], Ying Lu [3], Tarek Abdelzaher [4], Sang Son, Gang Tao

Department of Computer Science, University of Virginia, 151 Engineer's Way, Charlottesville, VA 22904-4740, United States

Available online 13 November 2006

Abstract

The resource management in distributed real-time systems becomes increasingly unpredictable with the proliferation of data-driven applications. Therefore, it is inefficient to allocate the resources statically to handle a set of highly dynamic tasks whose resource requirements (e.g., execution time) are unknown a prior. In this paper, we build a distributed real-time system based on the control theory, focusing on the computational resource management. Specifically, this work makes three important contributions. First, it allows the designer to specify the desired temporal behavior of system adaptation, such as the speed of convergence. This is in contrast to previous literature, specifying only steady-state metrics, e.g. the deadline miss ratio. Second, unlike QoS optimization approaches, our solution meets performance guarantees with no accurate knowledge of task execution parameters – a key advantage in a poorly modeled environment. Last, in contrast to ad hoc algorithms based on intuition and testing, we rigorously prove that our approach not only has excellent steady state behavior, but also meets stability, overshoot, and settling time requirements.
Published by Elsevier Inc.

Keywords: Real-time; Feedback control; Quality of service; Scheduling

1. Introduction

Distributed real-time systems are widely used in highly dynamic environments where the resource requirements are open, fluctuating and not amenable to the traditional worst-case real-time analysis. For example, a web farm can be used to distribute time-sensitive contents such as movies and video clips. They need to handle a changing number of requests with significantly different resource requirements that are unknown beforehand. In a stock market, a system needs to actively push real-time stock updates at various interval to a group of users. The number of users served by a server can change quickly over time. Although these systems differ significantly in term of applications, they all operate in open environments where both workloads and available resources are difficult to predict. Monitoring and feedback control are needed to meet performance constraints. Several difficulties are observed in dynamic resource management in these systems. One main difficulty lies in their data-dependent resource requirements, which cannot be predicted without interpreting input data. For example, the execution time of an information server (a web or database server) heavily depends on the content of requests, such as the particular web page requested. A second major challenge is that these systems have highly uncertain arrival workloads; it is not clear how many users will request some resource in the web. A third challenge involves the complex interactions among many distributed sites, often across an environment with poor or unpredictable timing behavior. Consequently,

* Corresponding author. Tel./fax: +1 612 6261281.
E-mail address: tianhe@cs.umn.edu (T. He).
[1] Department of Computer Science & Engineering, University of Minnesota, United States.
[2] Department of Computer Science & Engineering, Washington University in St. Louis, United States.
[3] Department of Computer Science & Engineering, University of Nebraska, Lincoln, United States.
[4] Department of Computer Science, University of Illinois, Urban-Champaign, United States.

0164-1212/$ - see front matter Published by Elsevier Inc.
doi:10.1016/j.jss.2006.09.029

developing certain types of future real-time systems will involve techniques for modeling the unpredictability of the environment, handling imprecise or incomplete knowledge, reacting to overload and unexpected failures (i.e., those not expressed by design-time failure hypotheses), and achieving the required performance levels and temporal behavior. We envision a trend in real-time computing to provide performance guarantees without the requirement of fine-grained task execution models, such as those depending on the precise estimation of individual task execution times. We shall see the emergence of coarse-grained models that describe the aggregate behavior of resource requirements. Coarse-grained models are easier to obtain and they need not be accurately computed. These models are more appropriate for dynamic resource management in the presence of uncertainties regarding load and resources.

In this paper, we explore one such model based on difference equations. Unlike the more familiar queuing theory models of aggregate behavior, difference equation models do not make assumptions regarding the statistics of the load arrival process. Independent of the load assumptions, difference equation models are more suitable for systems where load statistics are difficult to obtain or where the load does not follow a distribution that is easy to handle analytically. The latter is the case, for example, with web traffic, which cannot be modeled by a Poisson distribution. Our solution has a basis in the theory and practice of feedback control scheduling. This is in contrast to the more common ad hoc resource management based on intuition and testing where it is very difficult to characterize the aggregate performance of the system and where major overloads and/or anomalous behavior can occur since these designs are not developed to avoid these problems.

2. The overview of DFCS architecture

Traditional real-time computing provides guarantees in avoidance of undesirable effects such as overload and deadline misses. They assume worst-case resource requirements known a priori. In contrast, in highly uncertain environments, the main concern is to design adaptation capabilities that handle uncertain effects dynamically and in an analytically predictable manner. To address this issue, we propose a framework called Distributed Feedback Control real-time Scheduling (DFCS). The framework is based on feedback control that incrementally corrects system performance to achieve its target in the absence of initial load and resource assumptions. One main performance metric of such a system is the quality of performance–convergence to the desired level. In our framework, the desired convergence attributes may be specified and enforced using mechanisms borrowed from control theory. These mechanisms have been applied successfully for decades in physical process control systems that are often non-linear and subject to random external disturbances. Before establishing our DFCS framework, we give an overview of the software system being controlled and describe the feed-back-control mechanism involved. Note that although we focus on computational resource management here, while the general methodology can be applied to other dynamic resource management as well.

We assume that the resource under investigation is a cluster of computing nodes connected via a network. Tasks arrive at nodes in unknown patterns. Each task is served by a periodically invoked schedulable entity (such as a thread) with each instance having a soft deadline equal to its period. The periodicity constraint is motivated by the requirements of real-time applications such as process control and streaming media. We abstract a typical dynamic system by two sets of performance metrics. The *primary set* represents metrics to be maintained at specified levels, for example, the deadline miss ratio of a server, or the desired altitude of an airplane. The *secondary set* represents negotiable metrics such as service quality. The objective of adaptation is to incur minimum degradation in secondary metrics while maintaining the primary metrics at their desired values. To represent multiple levels of degradation in secondary metrics, we assume that each task has several service levels of different qualities. For example, a task can execute for varying amounts of time with the quality of the results improving with greater execution time. The goal of our DFCS architecture is to maintain the primary performance metrics around their desired values. Unlike a centralized system, the dynamics of a distributed system manifest themselves on two different time-scales. Fast dynamics are observed on individual nodes, while slower dynamics are observed on the entire system. The fast dynamics arise from local load changes due to individual task arrivals and terminations, while the slower dynamics arise from changes in aggregate load distribution. Therefore, our feed-back architecture naturally includes two sets of control loops, local and distributed ones, each tuned to the dynamics of the corresponding scale. Each node in the distributed system has a local feedback control system (LFC) and possibly a distributed feedback control system (DFC). The distributed feedback controller is responsible for maintaining the appropriate QoS balance between nodes. The local feedback controller is responsible for tracking the global QoS set point set by distributed controller and ensuring that tasks that are admitted to this node have a minimum miss ratio and the node remains fully utilized. It is important to note that these two types of controllers form the main parts of the distributed resource management in the system, but they are not the entire system.

Now consider a few more details about the DFCS architecture as shown in Fig. 1. The distributed controller (DFC) commands a set of local controllers (LFC) via a QoS set point, termed as service level ratio(SLR). The local controller (LFC) manipulates its actuators to achieve this SLR set point. In this architecture, we let the primary performance metric be the *deadline miss ratio* (MR). Since zero deadline miss ratio of admitted tasks can be trivially satisfied if the admitted task set is empty, it is especially