Chapter 15

On Application Behavior Extraction and Prediction to Support and Improve Process Scheduling Decisions

Evgueni Dodonov
University of São Paulo – ICMC, Brazil

Rodrigo Fernandes de Mello
University of São Paulo – ICMC, Brazil

ABSTRACT

The knowledge of application behavior allows predicting their expected workload and future operations. Such knowledge can be used to support, improve and optimize scheduling decisions by distributing data accesses and minimizing communication overheads. Different techniques can be used to obtain such knowledge, varying from simple source code analysis, sequential access pattern extraction, history-based approaches and on-line behavior extraction methods. The extracted behavior can be later classified into different groups, representing process execution states, and then used to predict future process events. This chapter describes different approaches, strategies and methods for application behavior extraction and classification, and also how this information can be used to predict new events, focusing on distributed process scheduling.

INTRODUCTION

The knowledge of the application behavior allows predicting the application workload during its execution and forecasting distributed data accesses. In order to obtain such data, different strategies can be employed, varying from simple source code analysis, sequential access pattern extraction (Kotz & Ellis, 1993), history-based approaches (Gibbons, 1997; Smith, Foster & Taylor, 1998) and on-line behavior extraction methods (Senger, Mello, Santana, & Santana, 2005; Dodonov, Mello, & Yang, 2006).

It is possible to define two different strategies for application behavior extraction. The first approach
On Application Behavior Extraction and Prediction

consists of a static source code evaluation, where the behavior is evaluated in an empiric way, without real application execution. The second, also known as dynamic evaluation, consists of the application behavior evaluation during execution.

The static evaluation approach is elderly, being conceived by Church and Turing, and is well described by Fischer (1965). Originally, the technique was applied to finite state automata and Turing machines, aiming at detecting possible problems which could lead to deadlock states or improper application terminations.

With the evolution of computing systems, new static evaluation techniques were introduced. Among those techniques are the model verification method, which aims at reducing the application behavior to a formal representation (Schuster, 2003), and the abstract interpretation method (Loiseaux, Graf, Sifakis, Bouajjani, & Bensalem, 1995), which represents the application behavior using a series of finite state machines, characterizing different application behavior using independent automata states.

Dynamic behavior evaluation technique, on its turn, investigates the behavior during process execution, usually with aid of debugging or monitoring utilities. It can be further divided into continuous monitoring and event-based approaches (Jain, 1991).

Continuous monitoring techniques consist of periodic extraction of application characteristics, determining the current behavior by evaluating differences among each execution state. This technique can be easily employed, as no application modification is required. However, as the monitoring occurs on pre-determined intervals, it introduces a constant overhead. Besides, a disadvantage of this approach lies in its impreciseness – as the monitoring occurs on fixed intervals, it is not possible to determine the precise behavior on specific execution states.

The event-based technique evaluates the behavior by determining critical execution states and extracting application characteristics when such states are reached, resulting in a more precise behavior determination. The adoption of this technique is more difficult, as it requires previous knowledge of application functionalities in order to correctly determine critical execution states that are further used to extract its behavior. Besides, this approach usually requires source code instrumentation or the interception and interpretation of function calls.

Among the advantages of event-based approach is the lower execution overhead, as the behavior is only extracted on specific execution points (for example, on data transfers, or thread synchronization operations).

The area of behavior prediction has received a lot of attention over the last years, resulting in a series of comparisons among different approaches, motivating competitions such as the K. U. Leuven (Suykens & Vandewalle, 2000), Eunite (Chen, Chang, & Lin, 2004) and Santa Fe (Weigend & Gershenfeld, 1994). However, most of the researches are focused on generic chaotic time-series prediction, or posterior behavior reconstruction. The study on the applicability of behavior prediction techniques to support and improve the performance of distributed systems, combined with our previous researches in this field, has motivated us to write this chapter, aiming at outlining different approaches and strategies employed on the process behavior extraction, classification and prediction, and their usage in distributed scheduling, load balancing and data access anticipation.

This chapter is organized as follows: Section 2 overviews the evolution of application behavior extraction strategies. Section 3 describes different approaches and techniques for behavior classification and prediction, and their usage in distributed scheduling, load balancing and data access anticipation. Finally, Section 4 outlines practical applications of the prediction results in distributed environments. Finally, Section 5 summarizes this chapter.
Related Content

Architecture Exploration Based on Tasks Partitioning Between Hardware, Software and Locality for a Wireless Vision Sensor Node
[www.igi-global.com/article/architecture-exploration-based-tasks-partitioning/66057?camid=4v1a](www.igi-global.com/article/architecture-exploration-based-tasks-partitioning/66057?camid=4v1a)

The Socio-Technical Virtual Organisation
[www.igi-global.com/chapter/socio-technical-virtual-organisation/19343?camid=4v1a](www.igi-global.com/chapter/socio-technical-virtual-organisation/19343?camid=4v1a)

Adaptive Mobile Architecture with Utility Computing
[www.igi-global.com/chapter/adaptive-mobile-architecture-with-utility-computing/139854?camid=4v1a](www.igi-global.com/chapter/adaptive-mobile-architecture-with-utility-computing/139854?camid=4v1a)

Design Space Exploration Using Cycle Accurate Simulator
[www.igi-global.com/chapter/design-space-exploration-using-cycle-accurate-simulator/159040?camid=4v1a](www.igi-global.com/chapter/design-space-exploration-using-cycle-accurate-simulator/159040?camid=4v1a)