Available online at www.sciencedirect.com
ScienceDirect
The Journal of Systems and Software 79 (2006) 1661–1678
The Journal of Systems and Software
www.elsevier.com/locate/jss

Distributed dynamic slicing of Java programs

Durga P. Mohapatra [a], Rajeev Kumar [b,*], Rajib Mall [b], D.S. Kumar [b], Mayank Bhasin [b]

[a] Department of Computer Science and Engineering, National Institute of Technology, Rourkela, Orissa 769 008, India
[b] Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur WB 721 302, India

Received 21 February 2005; received in revised form 15 January 2006; accepted 15 January 2006
Available online 28 February 2006

Abstract

We propose a novel dynamic slicing technique for distributed Java programs. We first construct the intermediate representation of a distributed Java program in the form of a set of *Distributed Program Dependence Graphs* (DPDG). We mark and unmark the edges of the DPDG appropriately as and when dependencies arise and cease during run-time. Our algorithm can run parallely on a network of computers, with each node in the network contributing to the dynamic slice in a fully distributed fashion. Our approach does not require any trace files to be maintained. Another advantage of our approach is that a slice is available even before a request for a slice is made. This appreciably reduces the response time of slicing commands. We have implemented the algorithm in a distributed environment. The results obtained from our experiments show promise.

Keywords: Program slicing; Dynamic slicing; Program dependence graph; Debugging; Object-oriented program; Multithreading; Java; Distributed programming

1. Introduction

As software applications grow larger and become more complex, program maintenance activities such as adding new functionalities, porting to new platforms, and correcting the reported bugs consume enormous effort. This is especially true for distributed object-oriented programs. In order to cope with this scenario, programmers need effective computer-supported techniques for decomposition and dependence analysis of programs. Program slicing is one technique for such decomposition and dependence analysis. A program slice with respect to a specified variable v at some program point P consists of those parts of the program which potentially affect the value of v at p. The pair $\langle v,p\rangle$ is known as the *slicing criterion*. A static slice is valid for all possible executions of a program, while a dynamic slice is meaningful for only a particular execution of a program (Ashida et al., 1999; Binkley et al., 1996; Korel and Laski, 1988). Program slicing has been found to be useful in a variety of applications such as debugging, program understanding, testing and maintenance (Agrawal et al., 1993; Gallagher and Lyle, 1991; Goswami and Mall, 2002; Kamkar, 1993; Mall, 2003; Mund et al., 2002; Zhang et al., 2004).

Many real life object-oriented programs are distributed in nature and run on different machines connected to a network. The emergence of message passing standards, such as MPI, and the commercial success of high speed networks have contributed to making message passing programming common place. Message passing programming has become an attractive option for tackling the vexing issues of portability, performance, and cost effectiveness. As distributed computing gains momentum, development and maintenance tools for these distributed systems seem to gain utmost importance.

Development of real life distributed object-oriented programs presents formidable challenge to the programmer. It is usually accepted that understanding and debugging of distributed object-oriented programs are much harder compared to those of sequential programs. A typical

* Corresponding author. Tel.: +91 5122596074; fax: +91 5122597586.
E-mail addresses: durga@nitrkl.ac.in (D.P. Mohapatra), rkumar@cse.iitkgp.ernet.in (R. Kumar), rajib@cse.iitkgp.ernet.in (R. Mall).

0164-1212/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2006.01.009

nature of distributed programs, lack of global states, unsynchronized interactions among threads, multiple threads of control and a dynamically varying number of processes are some reasons for this difficulty. An increasing amount of effort is being spent in debugging, testing and maintaining these products. Slicing techniques promise to come in handy at this point. Through the computation of a slice for a message passing program, one can significantly reduce the amount of code that a maintenance engineer has to analyze to achieve some maintenance tasks. However, research attempts in program slicing area have focused attention largely on sequential programs. Many research reports addressed efficient handling of data structures such as arrays, pointers, etc. in the sequential framework. Attempts have also been made to deal with unstructured constructs like goto, break, etc. Although researchers have reported extension of the traditional concept of program slicing to static slicing of distributed programs, dynamic slicing of distributed object-oriented programs has scarcely been reported in the literature.

Any effective dynamic slicing technique for distributed object-oriented programs needs to address the important concepts associated with object-oriented programs such as encapsulation, inheritance, polymorphism and message passing. This poses new challenges during slice computation which are not encountered in traditional program slicing and render representation and analysis techniques developed for imperative language programs inadequate. So, the object-oriented features need to be considered carefully in the process of slicing.

We have already mentioned that object-oriented programs are often large. Therefore, to be practically useful in interactive applications such as debugging, program traces should be avoided in the slicing process. Maintaining execution traces become unacceptable due to slow I/O operations. Further, to be useful in a distributed environment, the construction of slices should preferably be constructed in a distributed manner. Each node in a distributed system should contribute to the slice by determining its local portion of the global slice in a fully distributed fashion.

Keeping the above identified objectives in mind, in this paper we propose an algorithm for computing dynamic slices of distributed Java programs. Though we have considered only Java programs, programs in any other language can be handled by making only small changes to our algorithm. We have concentrated only on the *communication* and *concurrency* issues in Java. Standard sequential and object-oriented features are not discussed in this paper, as these are easily found in the literature (Ashida et al., 1999; Larson and Harrold, 1996; Liang and Larson, 1998; Krishnaswamy, 1994; Umemori et al., 2003; Wakinshaw et al., 2002; Wang and RoyChoudhury, 2004). For example, Larson and Harrold (1996) have discussed the techniques to represent the basic object-oriented features. Their technique can easily be incorporated into our algorithm to represent the basic object-orientation features.

We have named our proposed algorithm *distributed dynamic slicing* (DDS) algorithm for Java programs. To achieve fast response time, our algorithm can run in a fully distributed manner on several machines connected through a network, rather than running it on a centralized machine. We use local slicers at each node in a network. A local slicer is responsible for slicing the part of the program executions occurring on the local machine.

Our algorithm uses a modified program dependence graph (PDG) (Horwitz et al., 1990) as the intermediate representation. We have named this representation *distributed program dependence graph* (DPDG). We first statically construct the DPDG before run-time. Our algorithm *marks* and *unmarks* the edges of the DPDG appropriately as and when dependencies arise and cease during run-time. Such an approach is more time and space efficient and also completely does away with the necessity to maintain a trace file. This eliminates the slow file I/O operations that occur while accessing a trace file. Another advantage of our approach is that when a request for a slice for any slicing criterion is made, the required slice is already available. This appreciably reduces the response time of slicing commands.

The rest of the paper is organized as follows. In Section 2, we present some basic concepts and definitions that will be used in our algorithm. In Section 3, we discuss the intermediate program representation: *distributed program dependence graph* (DPDG). In Section 4, we present our *distributed dynamic slicing* (DDS) algorithm for distributed object-oriented programs. In Section 5, we briefly present an implementation of our algorithm. In Section 6, we compare our algorithm with related algorithms. Finally, Section 7 concludes the paper.

2. Basic concepts

Before presenting our dynamic slicing algorithm, we first briefly discuss the relevant features of Java. Then, we introduce a few basic concepts and definitions that would be used in our algorithm. In the following discussions and throughout the rest of the paper, we use the terms a *program statement*, a *node* and a *vertex* interchangeably.

2.1. Concurrency and communication in Java

Java supports concurrent programming using threads. A thread is a single sequential flow of control within a program. A thread is similar to a sequential program in the sense that each thread also has a beginning, an execution sequence and an end. Also, at any given time during the run of a thread, there is a single point of execution. However, a thread itself is not a program; it cannot run on its own. To support thread programming, Java provides a *Thread* class library, which defines a set of standard operations on a thread such as *start*( ), *stop*( ), *join*( ), *suspend*( ), *resume*( ) and *sleep*( ), etc. (Naughton and Schildt, 1998).