J-Orchestra: Automatic Java Application Partitioning (a Demonstration Proposal)

Eli Tilevich and Yannis Smaragdakis
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332
{tilevich, yannis}@cc.gatech.edu
http://j-orchestra.org

Abstract

This demonstration will present J-Orchestra [1] -- an automatic partitioning system for Java programs. J-Orchestra takes as input a Java application in bytecode format and transforms it into a distributed application, running across multiple Java Virtual Machines. To accomplish such automatic partitioning, J-Orchestra uses bytecode rewriting to substitute method calls with remote method calls, direct object references with proxy references, etc. The partitioning is performed without programming and without making any modifications to the JVM or its standard runtime classes. The main novelty and source of scalability of J-Orchestra is in its approach to dealing with unmodifiable code (e.g., Java system classes). The approach consists of an analysis algorithm to determine what references can leak to what parts of unmodifiable code, and a rewrite algorithm that maintains the references in the right form for the code that manipulates them.

Introduction

The focus of distributed computing has been shifting from "distribution for parallelism" to "resource-driven distribution", with the resources of an application being remote to each other or to the computation. Because of this shift, more and more distributed applications need to be adapted for distributed execution. Examples abound. A local database grows too large and needs to be moved to a powerful server, becoming remote to the rest of the application. An application wants to redirect its output to a superior remote graphical screen or to receive input from a remote digital camera. A desktop application when executed on a PDA might not find all the referenced APIs and their corresponding hardware resources available locally and wants to access them remotely.

All the aforementioned scenarios give rise to application partitioning [2]. Application partitioning is the task of splitting up the functionality of a centralized application into distinct entities running across different network sites. A standard way to accomplish such partitioning is to modify the source code of the original application by hand to use a middleware mechanism. This approach is tedious, error prone, and often simply infeasible due to the unavailability of source code, which is usually the case for commercial applications. We use an alternative approach that entails using a tool that under human guidance handles all the tedious details of distribution. This relieves the programmer of the necessity to deal with middleware directly and to understand all the potentially complex data sharing through pointers. Our tool, J-Orchestra, operates on binary (Java bytecode) applications and enables the user to determine object placement and mobility to obtain a meaningful partitioning. The application is then re-written to be partitioned automatically and different parts can run on different machines, on unmodified versions of the Java VM. For a large subset of Java, the resulting partitioned application is guaranteed to behave exactly like its original, centralized version. The requirement that the VM not be modified is important. We do not want to change the runtime, both because of deployment reasons (it is easy to run a partitioned application on a standard VM) and because of complexity reasons (Java code is platform-independent but the runtime system has a platform-specific, native-code implementation).

Demonstrating J-Orchestra's Partitioning

For this demonstration, we have prepared several centralized applications that run on a single JVM and use some system resources such as sound, graphics, etc. Among them are a graphical demo of the Java speech API (the user selects parameters and a sound synthesizer composes phrases), PowerPoint controller (a small Java GUI application that controls MS PowerPoint through its COM interface), an application for monitoring server load and displaying real-time graphical statistics, and some other graphical demos. All of the above will be partitioned in a client-server model, where the I/Opart of the functionality (graphics, text, etc.) is displayed on a client machine, while processing or execution of commands takes place on a server. Our client machine is a hand-held iPAQ PDA, running Linux. This environment is good for showcasing the capabilities of J-Orchestra—even relatively uninterest-

ing centralized applications become exciting demos when they are automatically turned into distributed applications, partly running on a hand-held device that communicates over a wireless network with a central server.

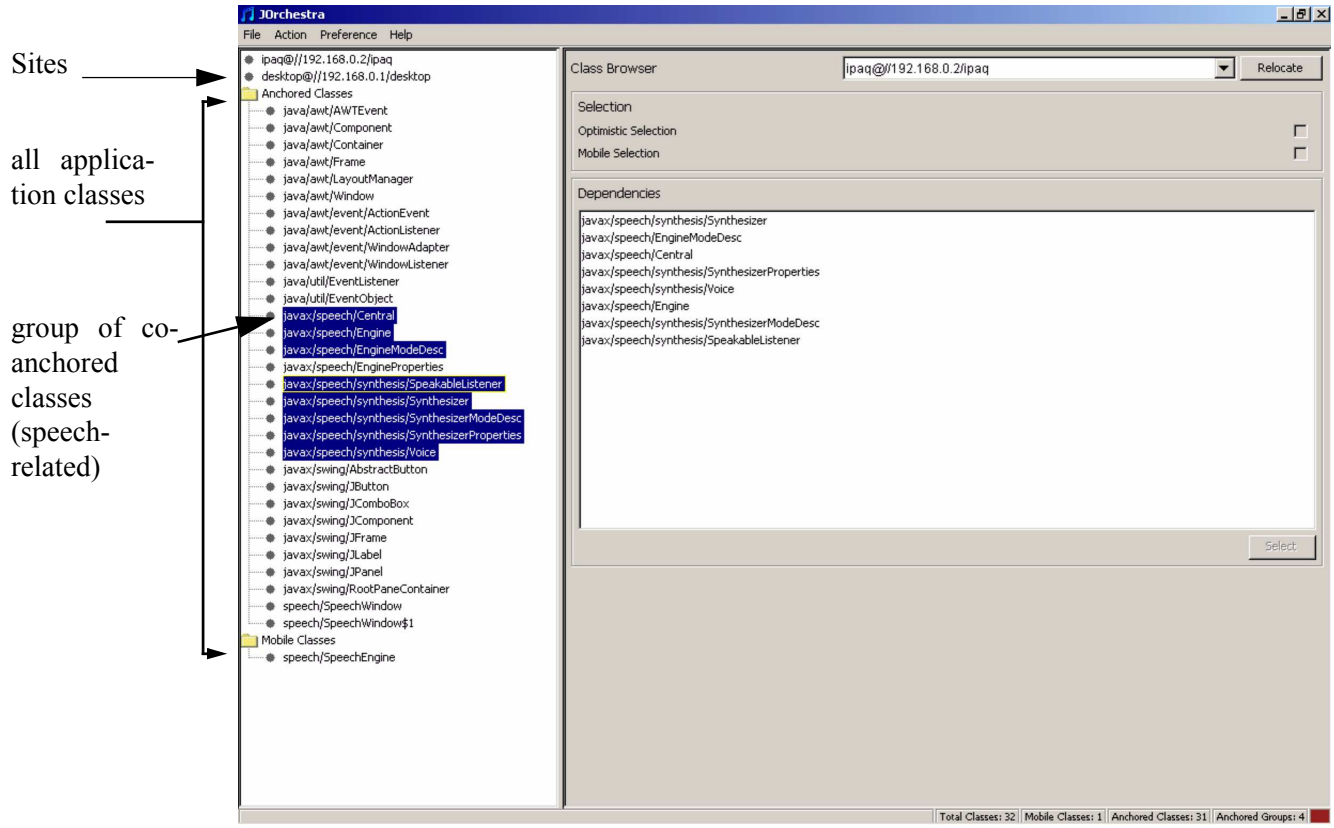


Figure 1. Example user interaction with J-Orchestra. An application controlling speech output is partitioned so that the machine doing the speech synthesis is different from the machine controlling the application through a GUI.

Availability

J-Orchestra is freely distributed under the LGPL. Check http://j-orchestra.org for details.

References

[1] Eli Tilevich and Yannis Smaragdakis, "J-Orchestra: Automatic Java Application Partitioning", *European Conference on Object-Oriented Programming (ECOOP)*, Malaga, June 2002.

[2] Eli Tilevich and Yannis Smaragdakis, "Automatic Application Partitioning: the J-Orchestra Approach", *8th ECOOP Worshop on Mobile Object Systems*, Malaga, June 2002.

J-Orchestra: Automatic Java Application Partitioning

http://j-orchestra.org

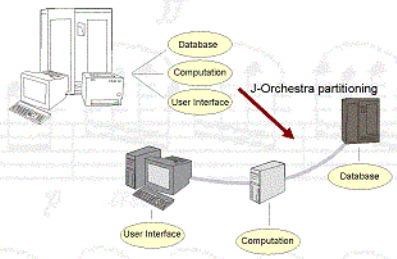
Eli Tilevich and Yannis Smaragdakis

{tilevich, yannis}@cc.gatech.edu

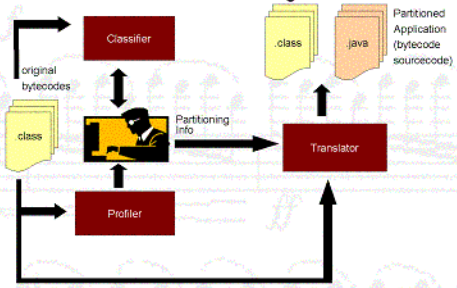
Center for Experimental Research in Computer Systems (CERCS)

Georgia Tech

Motivation for Application Partitioning



J-Orchestra Partitioning Process



Overview

•Take as input a Java application in bytecode format and transform it into a distributed application, running across multiple Java Virtual Machines.

• Enable the user to determine object placement and mobility to obtain meaningful partitioning.

• The main novelty is in dealing with *unmodifiable* code (e.g. Java system classes).

Challenges of Automatic Partitioning

• Handling the engineering complexity of partitioning realistic Java applications.

• Providing convenient GUI tools to enable the true "no-programming" style of partitioning.

• Achieving good performance for automatically partitioned applications.

J-Orchestra's Solutions

• Can partition *any* Java program allowing *any* application object to be placed on *any* machine, regardless of how application objects access each other and Java system objects.

• Can ensure good *performance* by employing *profiling*, *static analysis*, and enabling *object mobility*

GEORGIA

Georgia Tech College of Computing

cercs