THE SUMMARY OF Ph. D. DISSERTATION

Major School of Science for Open and Environmental Systems	Student Identification Number	SURNAME, Firstname CHIBA, Yuji

Title

A Study on Improving Performance of Java Applications through Translating to C Language

Abstract

Java is an object-oriented programming language with many advantages such as high productivity, but its performance is not always good. This is because general-purpose processors cannot execute a Java application directly. A Java application is written in intermediate code and needs software such as an interpreter for execution, but the performance of an interpreter is not good.

Compilers for Java can improve the performance by compiling the intermediate code into machine code and have the processor execute Java applications directly. But development of compilers costs much more than development of interpreters. One cost-effective way is to develop a translator from Java to C language, which we call Java2C translator, and then apply an existing C compiler that can conduct many kinds of traditional optimizations.

This paper shows problems that lie in the development of Java2C translators, and proposes solutions for them. The most important problem is that the machine code generated using the Java2C translator may not be usable at runtime. Because a Java2C translator translates the intermediate code before execution of Java application, the machine code generated using a Java2C translator becomes invalid if the intermediate code is updated after the translation. The invalid code must not be used for execution.

Java2C translators so far have neglected this problem and used machine code even when they are invalid. Thus, some Java applications have not run correctly when translated using such Java2C translators. This paper proposes a solution for this problem. Our solution confirms the validity of machine code at runtime before using them. We use an interpreter for execution until the confirmation, but the overhead for interpreter execution is not small. Thus, we also propose two optimizing techniques to reduce the overhead. One of the optimizations removes class initialization tests and the other is an implementation technique for virtual calls.

The result of applying SPECjvm98 benchmarks showed that removal of class initialization tests improve performance by 45% on average and the optimizing technique for virtual calls improves performance up to 9.8%.