発表概要

Javaにおける例外処理の実行時情報を利用した最適化

廣澤 健[†] 片岡正樹[†] 古関聰[††]
小松秀昭[††] 深澤良彰[†]

Java の言語仕様では，例外が発生する可能性がある部分を try ブロック内に記述し，例外発生時の処理を catch ブロック内に記述する．try ブロック中にメソッドの呼び出しがあった場合は，例外の発生時に JVM がメソッドのコールスタックを呼び出し順と逆順に参照して，適切な catch ブロックを検索するという処理を行う．このような例外処理仕様は，プログラムの可読性を高めるものとして支持されている反面，その実現のコストは小さくない．従来の最適化手法として，プロファイリングにより頻繁に実行される throw-catch 対を発見し，この部分をインライン展開し，インライン展開した部分中の throw 命令を jmp 命令で代替することにより，例外ハンドラ検索を高速化するというものがある．しかし，インライン展開には，コードサイズの増大によって，コンパイル時間や，インストラクションキャッシュミスが増加するという問題がある．我々の提案する手法では，まず，プロファイリングにより例外が頻発する部分を見つけ，この部分を最適化のターゲットとする．次に，ターゲットの実行時には，対象となる例外の catch ブロックのアドレスを専用のスタックに積み，例外発生時に該当する catch ブロックを専用のスタックから探すことで，検索を高速化する．本手法によりサンプルプログラムで最大約 7.0%の高速化を得た．

Optimization for Exception Handling in Java Using Exception Profile

Takeshi Hirosawa,[†] Masaki Kataoka,[†] Akira Koseki,[††]
Hideaki Komatsu[††] and Yoshiaki Fukazawa[†]

In the Java language specifications, a program segment where exceptions may occur have to be described in a try block and exception handling program have to be described in a catch block. In order to implement this mechanism, the Java Virtual Machine (JVM) searches the suitable catch block referring the call stack in the reverse order that the methods have been called. These exception handling specifications make us to read program easier, however its execution cost is not low. Previous work shows a new exception handling mechanism where the exception path is inlined after detecting throw-catch pairs which are executed frequently by profiling. Then the throw instruction is replaced with the jmp instruction. But the program size growth by the inlining causes instruction cash misses more often and makes a compile time longer. In this presentation, we suggest a new optimization technique of exception handling in Java. At first, we find the frequent executed exception path by profiling as optimizing target. Then, in execution time, we push the address of the target catch blocks on the exception stack. When the exception has occurred, we can detect the correspond catch block faster by searching the exception stack. We obtained the experimental results, it improved performance of exception intensive program by up to 7.0% without affecting performance of exception-minimal programs at all.

（平成 18 年 1 月 17 日発表）

† 早稲田大学大学院理工学研究科
Graduate School of Science Engineering, Waseda University
†† 日本 IBM 株式会社東京基礎研究所
Tokyo Research Laboratory, IBM Japan Ltd.