発表概要

コード書換えによる動的メソッド呼び出しの直接 Devirtualization

石崎一明† 安江俊明†
川人基弘† 小松秀昭†

本発表では，Java などの動的クラスロードをともなう言語において，実装が容易な動的メソッド呼び出しの直接 devirtualization 手法を提案する．本手法では，コンパイル時に動的メソッド呼び出しに対して直接 devirtualization されたコードとメソッドがオーバライドされた場合に実行する動的メソッド呼び出し，の 2 種類のコードを生成する．最初は前者を実行し，オーバライドが起きたときにコードを書き換えて後者を実行する．本手法では，コード書換えによって直接 devirtualization されたコードを無効化するので，脱最適化のような再コンパイルのための複雑な実装が不要である．一方，再コンパイルを不要にするためにコンパイル時に 2 種類のコードを用意するので，制御フロー上に合流点が生成される．一般に制御フローの合流点はコンパイラの最適化を妨げるが，本発表では合流点が存在しても十分な最適化を可能にする手法を示す．さらに，本手法と他の devirtualization 手法を組み合わせて Java の Just-In-Time コンパイラに実装し，評価を行った．その結果，devirtualization を行わない場合に比べ，SPECjvm98 と SPECjbb2000 において 0〜181%（平均 24%）性能を改善できることを示す．

A Direct Devirtualization Technique with the Code Patching Mechanism

Kazuaki Ishizaki,† Toshiaki Yasue,† Motohiro Kawahito† and Hideaki Komatsu†

This presentation presents a direct devirtualization technique for a language such as Java with dynamic class loading. The implementation of this technique is easy. For a given dynamic method call, a compiler generates the inlined code of the method, together with the code of making the dynamic call. Only the inlined code is actually executed until our assumption about the devirtualization becomes invalidated, at which time the compiler performs code patching to make the code of dynamic call executed subsequently. This technique does not require the complicated implementation such as deoptimization to recompile the method that is active on stack. Since this technique prevents some optimizations across the merge point between the inlined code and the dynamic call, we have further more proposed optimization techniques effectively. We made some experiments to understand the effectiveness and characteristics of the devirtualization techniques in our Java Just-In-Time compiler. In summary, we improved the execution performance of SPECjvm98 and SPECjbb2000 by ranging from 0% to 181% (with the geometric mean of 24%).

（平成 14 年 1 月 29 日発表）

† 日本 IBM 株式会社東京基礎研究所
Tokyo Research Laboratory, IBM Japan Ltd.