早稲田大学大学院 理工学研究科

# 博 士 論 文 概 要

## 論 文 題 目

# Design and Evaluation of Dynamic Optimizations for a Java Just-In-Time Compiler

# Java Just-In-Time コンパイラにおける動的最適化の設計と評価

申 請 者

| | |
|---|---|
| 菅沼 | 俊夫 |
| Toshio | Suganuma |


2007 年 5 月

本論文は、Java 言語の動的な特性に基づいた最適化、特に実行時プロファイルを用いた高速化手法の設計、実装および評価に関する研究をまとめたものである。

Java 言語は発表から約 10 年が経ち、ウェブサービスをはじめシステムの基幹部分を含めた様々な分野で使われるようになった。このように実用的なレベルに発展した裏には、Java 処理系自体の性能向上が必須であった。Java は、クラスローディングやリフレクションなどの動的な特性を持つ言語であり、多くの処理系では Just-In-Time (JIT)コンパイラを用いて、バイトコードから実行環境のバイナリーコードへ動的にコードの変換および最適化を行っている。

JIT などの動的コンパイラは、コンパイル作業がプログラムの実行中に行なわれ、またプログラム全体を見通す事ができないため、従来の静的コンパイラに比べ高価な最適化実行が困難である。すなわち実行性能の高いコードをできるだけ短いコンパイル時間で生成するという、相反する要求をいかに扱い解決するかが動的コンパイラにおける課題となる。しかし同時に、動的コンパイラには、実行中のプログラムからプロファイル情報を取得しそれを最適化に利用することができるという、静的コンパイラにはない優位性を持つ。

動的コンパイラにおけるプロファイル情報の利用に関しては、大きく二種類に分けられる。第一は実行中のホットスポットを発見し、プログラム中のどの部分を最適化すればよいかの指標とするものである。これにより、プログラム中の全てのメソッドを最適化する事なく、性能向上に最も寄与する部分にのみ最適化のための資源を投入する事が可能となる。この選択的最適化の手法は今日の先端的な Java 処理系の多くで採用されている。第二の方法は、実行中のプログラムの振る舞いを基に最適化のより効果的な適用を図るものである。これは軽量な実行時プロファイルの収集やその情報に基づく最適化など、第一の方法に比べより高度な仕組みが必要となるが、実行中の挙動に適応した最適化が可能となり、より高い性能レベルを期待することができる。反面その実用上の困難さから、これまで製品レベルの処理系では行われてこなかった課題である。

本論文では、これら二つの方法を用いてプロファイル情報を利用し最適化を行った結果を詳細に述べる。実装・評価はすべて、IBM 社製 Java JIT コンパイラを使用して行った。本論文の貢献は大きく分けると次の 3 点である。

本論文の第一の貢献は、動的最適化フレームワークの設計・実装と評価である。このシステムはインタープリタと再コンパイルフレームワークから成る複数レベルの実行モードを持つ。プログラムはまずインタープリタによって実行が開始される。各メソッドの実行回数やループの繰り返しの回数をカウントし、ある閾値を超えると JIT コンパイラを起動する。インタープリタと JIT が生成したコード間で互いにメソッド呼び出しが可能である。動的コンパイラには三段階の最適化レベルがあり、それぞれコンパイル時オーバーヘッドと生成されるコードの実行性能に関して異なるトレードオフを持つ。インタープリタと合わせた四つの実行

モードは、段階的なトレードオフレベルを備えた実行環境を提供する。

このシステムの特徴として、軽量かつ高い信頼度を持つオンラインプロファイル機能があげられる。プロファイル対象の実行モードの性能に対する敏感度や収集するプロファイル情報の性質により、三種類のプロファイル手法を使い分けている。第一は先述したインタープリタ実行におけるカウンタプロファイリング、第二はコンパイルされた実行コードに対するサンプリングプロファイリング、そして第三にインストゥルメントに基づくプロファイリングである。第一・第二はプログラム中のホットスポットを検出して実行モードを移行させるものであり、第三はそのホットスポットに対して、より詳細な挙動を収集し適応的な最適化を施すために使用される。すなわちシステムのフレームワークとして、最適化のための前述した二種類のプロファイル情報の利用が可能となっている。

このシステムの設計・実装および詳細な評価を行った。実験では全コンパイルを前提とした従来方式との比較を、実行性能とコンパイル時間などのオーバーヘッドを測定して行った。その結果我々の方式は、種々のアプリケーションを使ったプログラムの立ち上がりでは性能が約 2.7 倍向上、コンパイル時間は 1/5 以下に削減、また SPEC benchmark を用いたプログラムの定常状態では性能は同等、コンパイル時間は約 1/2 に削減と、全体として大きく優れている結果が得られた。また我々の方式は、プログラム実行中に挙動が変化しホットスポットが変わった場合においても、変化に追従し適応的にコンパイル対象を検出することを実証した。このフレームワークは現在でも、IBM の商用 Java 処理系で使われている。

本論文の第二の貢献は、上で述べた動的最適化フレームワークの上に構築した、プロファイル情報を使った最適化の設計・実装と評価である。ここでは二種類の最適化（メソッドインライニングとコード特殊化）を取り上げた。インライニングでは、プログラム中のホットスポットのメソッドに対して、呼び出し元のアドレス分布とその呼び出し頻度のプロファイル情報をある一定期間収集する。この情報から部分コールグラフを作成し、どのメソッドをどのコールパスに沿ってインラインすれば実行性能を最大化できるかを判定し、インライン方法の指定付きで再コンパイルする。アクセッサなどの非常に小さなメソッド以外はヒューリスティックを一切用いず、全てプロファイル情報に基づくインライン方式とした場合、実行性能の上でもコンパイルオーバーヘッドの面でも大きく向上することを示した。実験ではヒューリスティックを用いた従来方法に比べ、実行性能は最大 7%(平均 2%)向上、コンパイル時間は最大 65%(平均 40%)削減することができた。

コード特殊化とは、実行中の不変な又は稀にしか変化しない変数を検出し、これを利用した最適化を行うテクニックである。この際に重要な点は、プログラム中のどの変数が不変であると扱えれば何の最適化が可能となり、その結果どの程度の性能向上が見込まれるのかを正しく判断することである。この機能なしではプロファイル収集を行う対象変数を絞ることができず、プロファイルコストの大

きさにより実用化は不可能となる。我々はデータフローを使ったインパクト解析と呼ぶ手法を考案し、一段階前のコンパイル時に最適化インパクトの高い変数リストを作成し、再コンパイル時にそれらの変数を対象に値プロファイルをとることによって、最小のプロファイルコストで最大の性能向上を得る方法を開発した。コード特殊化により性能は最大 3%(平均 1%)向上する一方、コンパイル時間の増加は最大 30%(平均 10%)に抑えることができた。

本論文の第三の貢献は、領域ベースのコンパイル手法の設計・実装と評価である。領域コンパイルは元来、命令レベル並列用の静的コンパイラのため提案された手法であり、コンパイラがオフラインで収集されたプロファイル情報に基づいてプログラムを分割しコンパイル単位を決定する事で、より多くの最適化やスケジューリングの機会を得ようと考案されたものである。我々は動的コンパイラ環境においてこの概念が特に有効であると着目し、独自の考察・開発を行い実用化できる事を示した。まず実行時プロファイルとヒューリスティックを組み合わせ、コード中の実行頻度の低いパスをデータフロー解析によって求め、これをコンパイル対象から削除することによりメソッド内の領域選択を行った。次にこのメソッド単位の領域選択をインライニング処理に組み込む事でコンパイル領域を決定し、再コンパイルと動的スタック変換を実装する事により実行時領域脱出の処理を行い、我々の動的最適化フレームワーク上に実現した。

領域コンパイル方式を採用する事により、コンパイル時オーバーヘッドの削減という直接的な利点の他に、多くの最適化がより効果的になるという間接的な利点が得られる。例えばインライニングでは、対象メソッドの実行頻度の少ない部分はインラインされず、また実行頻度の低い呼び出し元は削除されておりインラインされることはない。従ってこの分をより有効なインライン対象に振り向けることができる。またデータフローを使った種々の最適化においても、頻度の高いパスと低いパスが合流する点が無くなり、データフロー情報の伝播が不要に阻害される事なく、より有効な最適化機会が得られる。又オブジェクトの脱出解析など一部の最適化では、領域を認識させ領域脱出時の補正処理を加える事で、より有効に働くようにする事ができる。

実験の結果、領域コンパイルとその最適化による性能向上は最大 24%(平均 4%)、コンパイル時間の削減は最大 45%(平均 18%)という結果が得られた。また実行時に領域脱出し再コンパイルが行われるケースは一部のアプリケーションの一部のメソッドに集中しており、他の最適化と組み合わせる事で領域脱出の頻度を減らすことができ、さらに高い効果が期待できることが判明した。

以上、本論文の貢献を再度まとめると、1) インタープリタを含む段階的コンパイルと軽量プロファイルを用いた再コンパイルフレームワークの設計と評価、2) プロファイル情報とインパクト解析を使った最適化の設計と評価、および 3) 動的コンパイラにおける領域コンパイル手法の設計と評価、の 3 点となる。

# 研　究　業　績

| 種 類 別 | 題名、　発表・発行掲載誌名、　発表・発行年月、　連名者（申請者含む） |
|---|---|
| 論文 | ○ Toshio Suganuma, Toshiaki Yasue, Toshio Nakatani. A Region-Based Compilation Technique for Dynamic Compilers, *ACM Transaction of Programming Languages and Systems (TOPLAS)*, Vol.28, No.1, pages 134-174, January 2006.<br><br>○ Toshio Suganuma, Toshiaki Yasue, Motohiro Kawahito, Hideaki Komatsu, Toshio Nakatani. Design and Evaluation of Dynamic Optimizations for a Java Just-In-Time Compiler, *ACM Transaction of Programming Languages and Systems (TOPLAS)*, Vol.27, No.4, pages 732-785, July 2005.<br><br>○ Toshio Suganuma, Takeshi Ogasawara, Kiyokuni Kawachiya, Mikio Takeuchi, Kazuaki Ishizaki, Akira Koseki, Tatsushi Inagaki, Toshiaki Yasue, Motohiro Kawahito, Tamiya Onodera, Hideaki Komatsu, Toshio Nakatani. Evolution of a Java Just-in-Time Compiler for IA-32 Platforms, *IBM Journal of Research and Development*, Vol.48, No.5/6, pages 767-795, September 2004.<br><br>Toshiaki Yasue, Toshio Suganuma, Hideaki Komatsu, Toshio Nakatani. Structural Path Profiling: An Efficient Online Path Profiling Framework for Just-In-Time Compilers, *Journal of Instruction Level Parallelism*, Vol.6, pages 1-28, April 2004.<br><br>Kazuaki Ishizaki, Mikio Takeuchi, Kiyokuni Kawachiya, Toshio Suganuma, Osamu Gohda, Tatsushi Inagaki, Akira Koseki, Kazunori Ogata, Motohiro Kawahito, Toshiaki Yasue, Takeshi Ogasawara, Tamiya Onodera, Hideaki Komatsu, Toshio Nakatani. Effectiveness of Cross-Platform Optimizations for a Java Just-In-Time Compiler, In *Proceedings of the 18th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA'03)*, pages 187-204, October 2003.<br><br>安江俊明，菅沼俊夫，小松秀昭，中谷登志男．動的コンパイラにおける実行時経路情報の構造的収集手法の提案，情報処理学会論文誌：プログラミング，Vol.44, No.SIG15，2003.<br><br>Toshiaki Yasue, Toshio Suganuma, Hideaki Komatsu, Toshio Nakatani. An Efficient Online Path Profiling Framework for Java Just-In-Time Compilers, In *Proceedings of the 12th International Conference on Parallel Architectures and Compilation Techniques (PACT'03)*, pages 148-158, September 2003.<br><br>○ Toshio Suganuma, Toshiaki Yasue, Toshio Nakatani. A Region-Based Compilation Technique for a Java Just-In-Time Compiler, In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'03)*, pages 312-323, June 2003. |

# 研 究 業 績

| 種 類 別 | 題名、　発表・発行掲載誌名、　発表・発行年月、　連名者（申請者含む） |
|---|---|
| 論文 | ○ Toshio Suganuma, Toshiaki Yasue, Toshio Nakatani. An Empirical Study of Method Inlining for a Java Just-In-Time Compiler, In *Proceedings of the 2nd Java Virtual Machine Research and Technology Symposium (JVM'02)*, pages 91-104, August 2002.<br><br>○ Toshio Suganuma, Toshiaki Yasue, Motohiro Kawahito, Hideaki Komatsu, Toshio Nakatani. A Dynamic Optimization Framework for a Java Just-In-Time Compiler, In *Proceedings of the 16th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA'01)*, pages 180-194, October 2001.<br><br>Kazuaki Ishizaki, Motohiro Kawahito, Toshiaki Yasue, Mikio Takeuchi, Takeshi Ogasawara, Toshio Suganuma, Tamiya Onodera, Hideaki Komatsu, Toshio Nakatani. Design, Implementation, and Evaluation of Optimizations in a Just-In-Time Compiler, *Concurrency: Practice and Experience*, Vol.12, No.6, pages 457-475, May 2000.<br><br>○ Toshio Suganuma, Takeshi Ogasawara, Mikio Takeuchi, Toshiaki Yasue, Motohiro Kawahito, Kazuaki Ishizaki, Hideaki Komatsu, Toshio Nakatani. Overview of the IBM Java Just-in-Time Compiler, *IBM Systems Journal*, Vol.39, No.1, pages 175-193, January 2000.<br><br>Kazuaki Ishizaki, Motohiro Kawahito, Toshiaki Yasue, Mikio Takeuchi, Takeshi Ogasawara, Toshio Suganuma, Tamiya Onodera, Hideaki Komatsu, Toshio Nakatani. Design, Implementation, and Evaluation of Optimizations in a Just-In-Time Compiler", In *Proceedings of the ACM SIGPLAN Java Grande Conference*, pages 119-128, June 1999. |
| 講演 | Java Just-In-Time コンパイラにおける最適化とその評価，電子情報通信学会技術研究報告，CPSY-99-64，pages 17-24，1999.<br>石崎一明，川人基弘，今野和浩，安江俊明，竹内幹雄，小笠原武史，<br>菅沼俊夫，小野寺民也，小松秀昭. |
| その他（論文） | 水野謙，菅沼俊夫，石崎一明，古関聡，上田陽平，小松秀昭. 並列トランザクショナルアプリケーションのためのプログラミングフレームワーク，情報処理学会論文誌：プログラミング，（掲載決定）.<br><br>Toshio Suganuma, Hideaki Komatsu, Toshio Nakatani. Detection and Global Optimization of Reduction Operations for Distributed Parallel Machines", In *Proceedings of the 10th International Conference on Supercomputing (ICS'96)*, pages 18-25, May 1996. |
| その他（講演） | 郷田修，大澤暁，小松秀昭，菅沼俊夫，小笠原武史，石崎一明，中谷登志男. HPF コンパイラの実装と評価，情報処理学会研究報告，HPC-57-20，pages 115-120，1995. |

# 研　究　業　績

| 種 類 別 | 題名、　発表・発行掲載誌名、　発表・発行年月、　連名者（申請者含む） |
|---|---|
| その他(総説) | 小松秀昭, 菅沼俊夫, 小笠原武史, 石崎一明, 郷田修. SP2 のための HPF コンパイラにおける最適化技術, 情報処理, Vol.38, No.2, pages 100-104, 1997. |