# 動的最適化システムにおけるプロファイリングを利用した分岐の最適化
# Optimization of Branch using Profiling in a Dynamic Optimization System

飯島 祐子 † 古関 聰 ‡ 小松 秀昭 ‡ 深澤 良彰 †
Yuko Iijima Akira Koseki Hideaki Komatsu Yoshiaki Fukazawa

## 1. はじめに

分岐命令はプログラムにおいて必要不可欠な制御構造の要素であるが，パイプライン方式のプロセッサにおいては分岐命令は実行速度を低下させる要因でもある．分岐命令は命令パイプラインを破壊する潜在的な因子を有しているからである．そのため，分岐命令の最適化はプログラムの実行速度を向上させるのに非常に効率的な手法である．しかし，分岐命令の分岐先に関する情報は実行時にしかわからないため，静的に分岐命令を最適化するには限界がある．

そこで，プログラム実行時の情報を利用した動的最適化手法を利用して分岐の最適化を行う手法が提案されている．本手法では，動的最適化システムである **DAISY** [1] においてエッジプロファイリングを利用した効率的な分岐の最適化を行う．

**DAISY** とは，*PowerPC* のバイナリ・アプリケーションを入力すると一度 VLIW コードにコンパイルした上で，その VLIW コードをシミュレートするというシステムである．一度コンパイルしたバイナリコードを入力するので，すでにディストリビュートされているバイナリ・アプリケーションを最適化することもできる．このシステムにおいてプロファイリングと最適化を行う機能を実装し，評価を行った．

## 2. 研究背景

本手法のベースとなる技術である，動的最適化とプロファイリングの手法について述べる．

### 2.1 動的最適化

動的最適化とは，入力や環境に合わせてプログラムの実行時に最適化を行うことである．実行時にならないとわからない動的な情報を利用することで，静的には行うことのできない最適化をすることができる場合がある．

動的最適化が有効な例として，条件分岐命令に対する最適化がある．分岐命令はプログラムにおいて必要不可欠な制御構造の要素であるが，命令パイプラインと分岐命令との関係を考えた場合，分岐命令は命令パイプラインを破壊し，プログラムの実行速度を低下させる潜在的な因子を有している．したがって，分岐命令を最適化することは，プログラムの実行速度の向上にきわめて有効な方法であるが，分岐命令の分岐先に関する情報は，静的なプログラム解析からは得ることができないため，分岐命令を最適化することは困難である．しかし，プログラムの実行時の情報，たとえば，分岐命令の分岐先の実行頻度などの情報を得ることができれば，ほとんど実行されない分岐命令を除去したり，分岐条件の反転を行うなどの最適化をすることができる．

### 2.2 プロファイリング

分岐命令の最適化のために，プロファイリングを行う．本稿では，エッジプロファイリングという手法を用いる．エッジプロファイリングとは，プログラム実行中に分岐においてどちらの分岐先が何回実行されたかを測定する手法である．コントロールフロープロファイリングの代表的な手法で，実行頻度の高いパスを選択するという最適化の基礎となっている [2].

エッジプロファイリングを行う最も単純なアプローチは，各基本ブロックにカウンタを置くことである．カウンタの値は基本ブロックが実行されるごとに増加していく．

本手法では，エッジプロファイリングの結果に基づいて，分岐条件の反転等の最適化を行う．

## 3. システムの概要

本手法では，動的最適化システムにおいて実行可能コードを入力し，そのコードを一度コンパイルし，プログラムの実行させて分岐に関する情報を収集（エッジプロファイリング）し，分岐先に関する情報を分析する．分析の結果，プログラムの再コンパイルが必要になった場合には再コンパイルをして最適化を行い，実行コードを出力する．

システムの流れは以下のようになっている．

1. 実行可能ファイルを入力
2. コンパイル
3. 条件分岐命令の検出
4. 基本ブロックにカウンタを挿入
5. プロファイリング
6. 再コンパイルと最適化

**条件分岐命令の検出**

分岐点となる基本ブロックを検出する．図 1(a) のような CFG（コントロールフローグラフ）の場合では 1 番のノードが分岐点となる．

**カウンタの挿入**

条件分岐命令の含まれる基本ブロックの分岐先を示すエッジにカウンタを挿入する（図 2）．図 1 では，2 と 3 の基本ブロックを指すエッジに当たる．カウンタには，2，3 の各基本ブロックの実行回数を代入する．

**プロファイリング**

プログラム実行時に，2 と 3 のノードの実行頻度を測定する．カウンタにはあらかじめ閾値を設定しておき，いずれかのカウンタの値が閾値を超えた場合には，再コンパイル処理を行うルーチンを呼び出す．

†早稲田大学理工学部
School of Science and Engineering, Waseda University
‡日本 IBM（株）
IBM Japan, Ltd.

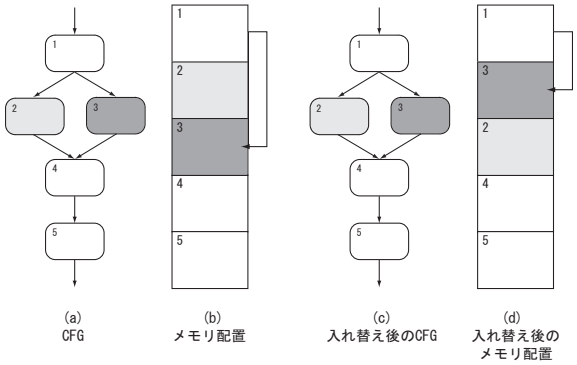


図 1: CFG とメモリ配置

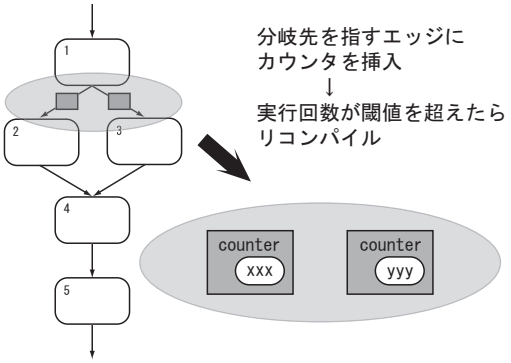


図 2: カウンタの挿入

**再コンパイルと最適化**

再コンパイル処理のルーチンが呼び出された場合は，分岐命令の最適化を行う．本システムでは，図 1(a) のような CFG の場合に，命令のメモリ配置は図 1 のようになっている．したがって，図 1(a) の 2 番のノードの位置に実行頻度の高い命令がくるように CFG を書き換えれば，キャッシュのヒット率が上がる．書き換え後の CFG とメモリ配置は，図 1(c)，(d) のようになる．

カウンタの値が閾値を超えた場合には，図 1(a) での 2 番と 3 番に当たるノードを入れ替える．ノードの入れ替えと同時に，分岐条件の反転も行う．このような最適化をすることで，実行頻度の高い命令がメモリ上で条件分岐命令のすぐ次に配置される (図 1(d)) ので，実行速度の向上が期待できる．

## 4. 実験

本手法を実装し，評価を行った．動的最適化システムに本手法を適用した場合と適用しなかたった場合とで，実行速度の比較をした．

### 4.1 実験方法

評価には動的最適化システムとして **DAISY** を使用した．**DAISY** にプロファイリング処理と再コンパイル処理を組み込み，オリジナルの **DAISY** との実行速度の比較を行った．

実験には分岐先の実行頻度に偏りがあるプログラム (branch1，branch2，branch3)，分岐先がさらに分岐しているプログラム (2branchs1,2branchs2) を使用した．branch1，branch2 のプログラムは，分岐の形は同じであるが分岐先の実行頻度が異なるものである．1 → 2 の順に then 部の実行頻度が高くなる．branch3 のプログラムは分岐中の命令数が branch1，branch2 のプログラムの 2 倍になっている．2branchs1，2branchs2 のプログラムも branch1，branch2 と同様に，分岐の形は同じで分岐先の実行頻度の異なるもので，1 → 2 の順に then 部の実行頻度が高くなる．branch1，2branchs1 のプログラムは，branch2，2branchs2 よりも二つの分岐先の実行頻度の偏りが大きくなっている．

### 4.2 実験結果

本手法適用前の **DAISY** における実行速度と本手法適用後の **DAISY** における実行速度 (単位：秒) と，本手法適用前の場合に対する本手法適用後の速度比を表 1 に示す．

表 1: DAISY における実行速度の比較

| プログラム | 本手法適用前 | 本手法適用後 | 速度比 |
|---|---|---|---|
| branch1 | 92.017 | 92.797 | 0.992 |
| branch2 | 102.907 | 102.457 | 1.004 |
| branch3 | 127.120 | 127.560 | 0.997 |
| 2branchs1 | 139.903 | 142.283 | 0.980 |
| 2branchs2 | 149.940 | 149.263 | 1.005 |

表 1 のように，本手法を適用した場合に実行速度の向上がみられる場合もあった．分岐命令の分岐先はブランチターゲットアドレスキャッシュに格納される．このキャッシュにヒットしなかった場合にはパイプラインがストールしてしまう．本システムで使用したプロセッサは *PowerPC* であるが，ブランチターゲットアドレスキャッシュの数は有限である．本手法を適用すると使用されない分岐先をこのキャッシュに格納しないので，キャッシュミスが起こりにくくなり，速度が向上したと考えられる．branch1，branch2，2branchs1，2branchs2 の結果にみられるように，分岐先の偏りが大きいほど本手法を適用した場合の効果は大きいことがわかった．

## 5. おわりに

本稿では，動的最適化システムにおいてプロファイリングを利用して効率的に分岐を最適化する手法を提案した．本手法を適用する前と適用した後での実行速度を比較した結果，若干ではあるが，本手法を適用した方が実行速度が速くなることがわかった．

本稿では，分岐命令の条件と分岐先を反転するという最適化手法をシステムに組み込んだが，さらに異なる最適化手法を組み込むことでより性能の向上が期待できる．

## 参考文献

[1] Erik R. Altman and Kemal Ebcioglu: “**DAISY** Dynamic Binary Translation Software” *IBM T.J. Watson Research Center*, 2000.

[2] Thomas Ball, Peter Mataga and Mooly Sagiv: “Edge Profiling versus Path Profiling: The Showdown” *Proceedings of the 25th ACM SIGPLAN-SIGACT Symposium on Principles of Programing Languages* pp. 134-148, Jan. 1998.