分岐確率と広域命令スケジューリング *

4T－3

林 正和　松山 学　堀田耕一郎 †

富士通株式会社‡

1 はじめに

1サイクルで複数命令実行できるRISCプロセッサのコンパイラにおいて、命令スケジューリングは、実行性能の向上にとって有効な手法である。我々は、情報処理学会第47回全国大会において、分岐命令を越えて命令を移動する広域命令スケジューリングの効果について発表した([1])。我々の採用した広域命令スケジューリング技法は、トレーススケジューリング([2])と呼ばれ、分岐が存在した場合は、より多く分岐する経路を、1つのスケジューリング範囲とする方法である。

今回は、コンパイラにおいて、広域命令スケジューリングの効果を高めるために、分岐する確率（以下、分岐真率と呼ぶ）を手動／自動でコンパイラに指定する方法を考案した。今回は、これらによって得られたデータを元に、分岐の真率と広域スケジューリングの効果の関係について述べる。

2 トレーススケジューリング

まず、トレーススケジューリングの概要を説明する。基本ブロック内の命令スケジューリングでは、命令列中に分岐命令が現れると、そこでスケジューリングの範囲が切れてしまうため、分岐が頻繁に現れると、命令を移動する範囲が狭くなってしまう。トレーススケジューリングは、分岐が現れると、より多く分岐するパスを予想し、そのパスをスケジューリング範囲に加えていくことによって、命令移動の範囲を広げ、命令スケジューリングの効果を高めることを狙ったものである。

ところが実際には、トレースを選択しなかった方のパスを実行することがある。その時には、分岐予測が外れたために、実行すべき命令が実行されない場合がありうる。従って、このような命令を分岐先／分岐の合流先にコピーする処理も行なわれ、実行結果の誤りを防いでいる。

*Branch Probability and Global Instruction Scheduling

†Masakazu HAYASHI, Manabu MATSUYAMA, KohIchiro HOTTA

‡FUJITSU LIMITED

3 分岐真率とトレーススケジューリング

2. で述べたように、トレーススケジューリングでは、補正コードを実行してしまうことによる実行性能のロスが存在する。従って、補正コードの実行頻度が多いと、それだけ実行性能も低下する。また、本コンパイラはレジスタ割り付け処理も、このトレースの選択順に優先的に行なうため、「トレースの選択をどれだけ正しく行なうことができるか？」が、トレーススケジューリングによって得られる実行性能に影響する。そのためには、コンパイラが分岐の真率を正しく把握する必要がある。

4 分岐真率の設定方法

コンパイラが、分岐真率を知る手段としては、大きく分けて2つの方法がある。1つは、コンパイラ自身で、分岐真率を判断してしまうものである。この手法で、1番簡単なものは、全ての分岐に対して、defaultの分岐真率を与えるという手段である。この手法は容易であるが、外れる確率も高く、そのためトレーススケジューリングの効果を十分に引き出すことができない場合がある。また、この手法の改良として、プログラム構造や、分岐の種別から、ヒューリスティックな手法を用いて、分岐するかどうかの判定を行なう方法も存在する。

もう1つは、ユーザー（プログラマ）や、プロファイル情報などコンパイラ外部からの手助けを借りる方法である。この手法は、多少の手間が必要であるが、コンパイラに分岐真率情報を与えるため、トレーススケジューリングの効果を引き出すことができる。ここでは、後者の方法を中心に述べる。

4.1 翻訳指示行で分岐真率を指定する方法

我々のコンパイラでは、以下の翻訳指示文を、プログラム上のif文の直前に記入することによって、分岐真率を指定することができる。ここで、Nは、分岐真率を百分率で示したものとする。

#pragma statement if N (C言語の場合)

!OCL IF (N) (Fortran言語の場合)

この手法の利点は、プログラマが予め分岐真率を予想できたり、既存のプロファイルのデータ等が存在している場合などには、プログラムを実行させる

表 1: 分岐真率情報の効果

	プログラム		
CASE	dhrystone	compress	mdljdp2
(1)	1.0	1.0	1.0
(2)	1.063	1.036.	1.000
(3)	1.074	1.060	1.017

CASE1: 分岐真率情報なし (1.0)
CASE2: 分岐真率情報をプログラム中に埋め込む
CASE3: 分岐真率情報をプログラムを実行して得る

ことなく、分岐真率を指定できるということである。しかし、一般的には実行頻度が高い場所に、IF 文に展開されるマクロが記述されていたり、IF 文に展開される switch 文が存在したり、IF 文の条件の組合せが複数存在したりするなど、正確な分岐真率を 指定できないプログラムが存在する。 このような場合は、全ての分岐命令に対して真率が指定できるわけではないため、本手法の効果が期待できない。また、全ての IF 文に対して、手修正でこの情報を入力することは、非常に手間がかかる作業となることなども、本手法の欠点であろう。

4.2 プログラムを実行させて分岐情報を得る方法

4.1 に示すように、翻訳指示行では全ての分岐命令に対して、分岐真率を指定することは難しい。そこで、1 度プログラムを実行させて各分岐命令に対して、そこに何回到達し、何回分岐したかの回数をカウントすることにより、分岐真率を求め、それを利用する仕組みが必要になってくる。我々のコンパイラでは、次のステップで個々の分岐真率を利用することができる。

1. プロファイル情報収集オプションで翻訳されたロードモジュールを実行することによって、個々の条件分岐に対し、taken/not taken をカウントし、その結果を情報ファイルに格納する。

2. プロファイル情報を利用するオプションが指定されたら、コンパイラは、上記で得られた情報ファイルを元に、各分岐命令の真率を計算し、スケジューラはその情報を利用して、トレースの選択を行なう。

5 実行結果

分岐真率を与えた時の実行性能を比較し、表 1 に載せた。分岐真率は、翻訳指示行によって指定した場合 (CASE2) と、自動的にプロファイル情報から設定する場合 (CASE3) の 2 つの方法で比較している。それぞれ、分岐真率を指定しない場合 (CASE1) を 1 とした時の相対比である。本コンパイラでは、様々な試行により、分岐真率の情報が与えられない場合は、ループの back edge は 90%、その他は 49% の確率で分岐するという処理にした。尚、測定したマシンは、SS10-41 (SuperSPARC 40MHz) である。

mdljdp2 において、この分岐真率の効果がほとんどなかったのは、このプログラムにおいて、最も実行されるループにおいて、コンパイラの予想がほとんど当たったためであると考えられる。

また、dhrystone や compress は共に、分岐真率を与えたほうが 4% ～ 8% ほど、性能が上がっていることがわかる。dhrystone の場合は、CASE2 と CASE3 で、性能向上率があまり変わらないが、これは、dhrystone は program が小さく、if 文に展開するマクロが存在しなかったために、プログラム上に分岐真率の情報を十分に埋め込むことができたためである。compress の場合は、ファイル内の全ての if 文には情報を埋め込んだが、マクロで分岐に展開される部分や、複合条件からなる IF 文がプログラム中に存在したために、CASE2 の方法では、実際の全ての分岐命令に対して分岐真率の情報が与えることができなかった。その結果が CASE2 と CASE3 の実行性能の差として現れたと考えられる。

6 まとめ

分岐の真率を、プログラムに指定する方法、プログラムを実行させた後に、自動的に指定する方法を用いた場合のそれぞれのトレーススケジューリングの効果について述べた。今回は、分岐真率をトレーススケジューリングのトレース選択に利用した場合に主眼をおいてその効果を見たが、分岐真率の情報は、その他の最適化においてもいろいろと利用できるものである。また現在では、コンパイラの分岐予測の技術も文献 [3] にあるように、着目・改良すべき点がいくつかある。

今後は、分岐真率情報のコンパイラの最適化における利用技術及び分岐予測技術の改良を目標として、研究・開発を続けていく所存である。

参考文献

[1] 松山 他, 「*RISC* プロセッサ向け広域命令スケジューリング」, 情報処理学会第 47 回全国大会, 1993

[2] John R. Ellis, *'Bulldog : A Compiler for VLIW Architecture'*, The MIT Press 1986

[3] THOMAS BALL and JAMES R. LARUS, *'Branch Prediction For Free'*, ACM SIGPLAN Conference on PLDI 1993 pp.300-313