62A-08　コンパイラによる値予測支援

飯塚 大介,　小沢 年弘 †,　坂井 修一,　田中 英彦
{iizuka,sakai,tanaka}@mtl.t.u-tokyo.ac.jp, ozawa@flab.fujitsu.co.jp†
東京大学大学院 工学系研究科,　富士通研究所 † *

1　はじめに

プログラム中に含まれるデータ依存関係を解決する手法として値予測がある。値予測はミスが起きると大きなペナルティーになるので、予測ミスをなるべく起こさないように、プロファイル等を用いて予測率の高い命令に対してのみ値予測を行なう方式が提案されている [1]。しかし、予測率が高い命令であっても、予測することによって利益が出なければ予測する意味がなく、逆に、予測率が低い命令であっても、当たることによる利益が予測ミスによるペナルティーを上回っているのであれば予測する意味があると考えられる。

さらに、プロファイルを用いた場合はプログラムの入力データが異なると予測率が変化してしまうため、どのような入力データに対しても、値予測を行なうことで効果のある命令を選択するには、コンパイラによって静的に解析を行なうのが適していると思われる。

また、効果のある命令全てについて予測を行なうと、値予測のためのハードウェア量が大きくなってしまう問題が生じるため、

最小限の予測で最大の効果を得るにはどのような命令を予測対象に選べば良いのかを調べることが重要となる。

本稿では、プロファイルを用いて予測率の高い命令を得た後に、それらの中から効果のある命令を選択して予測を行なうことで、予測率の高い命令全てを予測した場合とほぼ同等の速度向上が得られたことを示す。

最終的には、これ以外にも予測する命令の選択方式を求め、コンパイラによって静的に値予測する命令を抽出するための指標にする予定である。

2　値予測機構

値予測の機構には、主なものとして直前の実行結果と同じ値を返す last-value 測機構や、前々回と前回の実行結果の差から今回の値を予測する stride-value 予測機構等がある [2]。この他にも多くの予測機構が提案されているが、これら二つは値予測に必要なハードウェア量が少なく、利益が大きいために、本研究ではこれら二つの値予測機構を利用することにした。

* "Compiler Supported Value Prediction"

Daisuke Iizuka, Toshihito Ozawa †, Shuichi Sakai, Hidehiko Tanaka
University of Tokyo, Graduate School of Engineering,
Fujitsu Laboratry †

3　値予測する命令の選択

値予測を行なう命令について、まずは [1] のように、予測ミスによるペナルティーを最小限に抑えるため、予測率が 99% 以上となる命令をプロファイルから求め、この中から、速度向上に寄与する命令を選択した。

3.1　予測対象命令

予測対象となる命令は、通常のレジスタに値を生成する全命令とした。そのため、ロードストア命令のみを予測する場合よりも性能が向上すると思われる。

3.2　速度向上に寄与する命令の選択

値予測する命令 A と、その命令で生成される値を始めて使う命令 B があるとする。その際に、図 1のように命令 B が命令 A よりも先に実行される場合には値予測による効果がある。しかし、命令 B の実行に必要な全てのデータがすぐには揃わず、図 2のように命令 A の後に実行される場合には、命令 A は値予測する意味はないために選択から外すこととする。

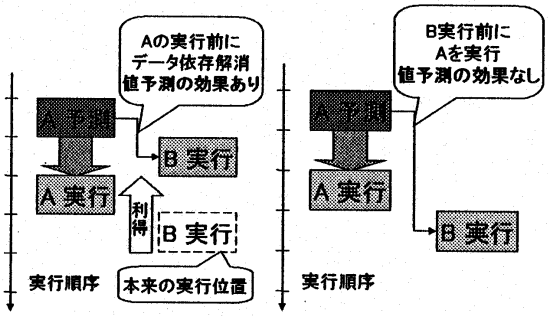

図 1: 効果がある場合

図 2: 効果がない場合

具体的には、同一の PC 上の命令 (静的な命令) について、以下の式を満たすようなものについてのみ予測を行なうこととした。

$$\overline{DestLatency} = \frac{\sum DestLatency}{\sum DestExistTimes} \quad (1)$$

$$\overline{ExecLatency} = \frac{\sum ExecLatency}{\sum ExecTimes} \quad (2)$$

$$\overline{DestLatency} \leq \overline{ExecLatency} + \alpha \quad (\alpha\text{はパラメータ}) \quad (3)$$

変数の説明:

- *DestLatency*
  値予測を行なった命令のデータが、値予測を行なわない命令に最初に使われるまでのレイテンシ
- *DestExistTimes*
  値予測で生成された命令が、値予測を行なわない他の命令で使われる場合の回数
- *ExecLatency*
  値予測したデータが使用可能になってから、実際に実行されるまでのレイテンシ
- *ExecTimes*
  値予測を行なう命令が実行された回数

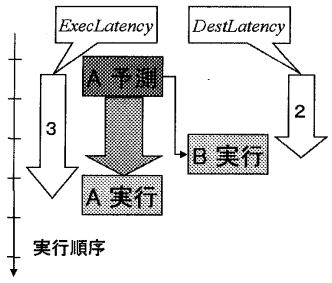

図 3: 変数の説明図

図3からわかるように、

$$DestLatency > ExecLatency + 1 \qquad (4)$$

の場合は、命令 A は値予測を行なわずに通常に実行しても、命令 B を実行するタイミングは変化しない。そのため、

$$DestLatency \leq ExecLatency + 1 \qquad (5)$$

となる命令のみ値予測を行なうこととすれば、値予測を行なう命令を減らすことができる。式3は式5から導いたものであるが、同一 PC 上の全ての命令が式5を満すとは限らないため、$\alpha = 1$ が最適であるとはいえず、それぞれのプログラムや、プログラム中のそれぞれの同一 PC の命令毎に最適な値があると思われる。

4 評価

前章で述べたような値予測する命令の選択を行なった場合、どの程度の速度向上が得られるのか評価を行なった。

評価環境として、リオーダバッファ以外の資源が無限大で、アウトオブオーダ実行を行なうプロセッサを仮定した。命令は全て1サイクルで終了し、ロード・ストアのアドレスもパイプラインの早い段階で予測され、依存関係のないロード・ストアものはアウトオブオーダ実行を行なうこととする。キャッシュは全てヒットするものとし、分岐予測は 100% 当たるものとする。

また、値予測が外れた場合はペナルティーとして5サイクルかかり、その命令が予測を始めた箇所から再度実行しなおすことにしてある。

ベンチマークは SPECint95 を使用し、実行コードは富士通研究所の C コンパイラ [3] が出力する SPARC のバイナリを用いた。上記プロセッサモデルを実装したシミュレータ上で、通常の実行、99% 以上当たる予測率の高い命令全てを予測対象とした値予測、予測率の高い命令から式3で $\alpha = 1$ を満たす命令のみ抽出した値予測の3つについて、SPECint95 のベンチマークを動作させて測定を行なった。結果の一部を以下に示す。

compress95 リオーダバッファ =128	通常実行	値予測	
		99%	選択
実行サイクル数 (ins.)	4.21M	2.60M	2.60M
速度向上率 (%)	100	159	159
静的な予測命令数	0	443	140
動的な予測命令数	0	32.24M	16.77M
サイクル当たりの予測数の平均	0	12.44	6.45
サイクル当たりの値予測最大数	0	57	19

go リオーダバッファ =128	通常実行	値予測	
		99%	選択
実行サイクル数	3.93M	2.46M	2.55M
速度向上率 (%)	100	160	154
静的な予測命令数	0	443	141
動的な予測命令数	0	36.69M	20.6M
サイクル当たりの予測数の平均	0	14.89	8.09
サイクル当たりの値予測最大数	0	81	74

5 まとめと今後の課題

値予測する命令を、効果のあるものだけ選択することで、動的な値予測命令数を減らしつつ、速度向上が得られることがわかった。

今後はさらに、どのような命令を値予測することで性能向上が計れるのかを調査し、コンパイラによりそのような命令を静的に解析できるようにする予定である。

謝辞

本研究を進めるにあたり、富士通研究所の木村康則氏に定期的に御助言を頂きました。深く感謝致します。

参考文献

[1] Feddy Gabbay, el al. Can Program profiling support value prediction? *MICRO*, pp. 270–280, December 1997.

[2] 吉瀬 謙二, 他. マルチレベル・ストライド値予測機構による命令レベル並列性の向上. *JSPP*, pp. 119–126, Jun 1999.

[3] 飯塚 大介 他. C コンパイラにおけるループ最適化の検討. *HPC 77*, pp. 65–70, Aug 1999.