コンパイラによる静的分岐予測

4T－4

細井聡　木村康則
（株）富士通研究所

1　はじめに

パイプライン計算機を効率良く実行するためには、できるだけ分岐のペナルティを軽減することが必要である。そのため、従来から様々な分岐予測に関する研究がなされてきた。整数系プログラムは、数値計算プログラムに比べて分岐命令が多く、したがって、分岐予測はよりいっそう重要となる。今回、整数系プログラムに対して、コンパイラがソース情報のみから、どのくらい正確に分岐予測を行なうことができるかを調査したので報告する。

2　ベンチマークプログラムとターゲットアーキテクチャ

ベンチマークプログラムとして、SPECint92 のうちの compress, xlisp, eqntott, espresso, gcc を用いて、分岐予測の正しさ（精度）を求めた。ターゲットアーキテクチャは、SPARC-V8 ベースのスーパースカラを想定した。

3　ヒューリスティックを用いた分岐予測

コンパイラは、ループなどの制御構造を把握したり、各分岐命令の周辺の状態をあらかじめチェックすることができる。したがって、動的な情報がなくても、ある程度は分岐予測を行なうことが可能である。静的分岐予測の研究としては、様々なヒューリスティックを用いた [1] がある。しかし、[1] は、

- ヒューリスティックの中には、特定のプログラムにしか有効ではなく、他に対しては予測が悪くなるものがある
- プログラムによっては、分岐方向を乱数によって決定している割合が大きい

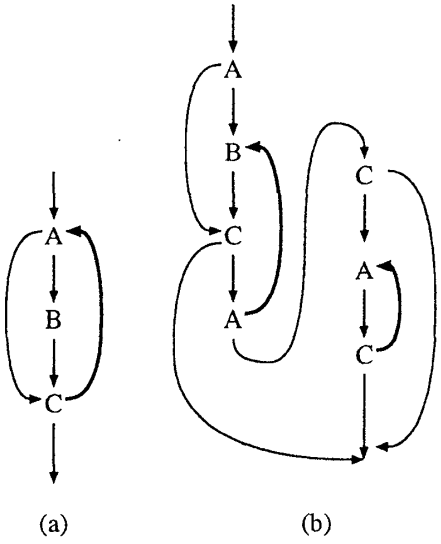

図 1: ループの構造変換

という問題点があった。そこで、我々は、できるだけ簡単かつ一般的なヒューリスティックのみでどのくらい効果があるかを調べることにした。我々が用いたヒューリスティックは以下の3つである。（これらのうち、ヒューリスティック1だけは [1] でも用いられている。）

1. 「ループが続く方向に分岐する」　すなわち、loop backward jump は taken, loop exit jump は not taken のことが多い。
2. 「例外処理関数の実行頻度は小さい」　abort (), _flsbuf () などの関数、および、関数名に abort, fail, error などの文字列を含んでいる関数のあるパスへは分岐しないことが多い。
3. 「ループ内分岐はどちらにも分岐する可能性がある」　ここでは、以下のような特殊な場合を考える。すなわち、

$$for(;;)\ if(q)\{...\}$$

のようなループにおいて、if 文の条件が成立することが多いとは限らない。このループの構造は、

Static Branch Prediction by Compiler
Akira Hosoi, Yasunori Kimura
FUJITSU LABORATORIES LTD.
1015, Kamikodanaka Nakahara-ku, Kawasaki 211, Japan

表 1: 分岐予測の精度

	Heu(%)	Default(%)	BTB(%)
compress	83.8	60.0	86.4
xlisp	76.6	58.4	87.5
eqntott	71.2	51.9	75.9
espresso	78.9	68.0	87.5
gcc	59.9	56.8	88.5

図 1(a) のように表現できるが、$A \rightarrow B \rightarrow C$ および $A \rightarrow C$ の 2 つのパスのうちのどちらを通ることが多いかは一概に言えない。ほとんど $A \rightarrow C$ を通るのに、$A \rightarrow B \rightarrow C$ であると予測すれば、予測がはずれる割合が多くなる。そこで、これを図 1(b) のように変更して、ループを 2 つ作る。これにより、2 つのパスのうちどちらが通ることが多い場合にも対応できるようになる。

（なお、ヒューリスティック 1 ～ 3 にあてはまらない分岐命令は、「後方分岐は taken, 前方分岐は not taken」と予測した。）

これらのうち、ヒューリスティック 1 は compress, eqntott、ヒューリスティック 2 は xlisp, compress、ヒューリスティック 3 は espresso に特に効果があった。

4 分岐予測の精度の比較

コンパイラによる静的分岐予測（Heu）を、

- Default: 単純な分岐予測としてよく用いられる手法で、後方分岐は taken, 前方分岐は not taken と予測
- BTB: 16KB(32 bytes block, 128 entry, 4-way, 2-bit prediction) の分岐先バッファを用いた動的分岐予測

と比較した結果を表 1に示す。これより、静的分岐予測の精度について以下のことがわかる。

- gcc 以外は、Default に比べて 10 ～ 20% 良い。
- compress や eqntott は BTB と同程度の精度が得られているが、他のプログラムでは 10% 以上 BTB より劣る。

BTB と 静的分岐予測との差が生じる理由

実行トレースを解析すると、1 つ 1 つの分岐命令は、ほとんど taken となるか、あるいはほとんど not taken となるかのいずれかであることが多いことがわかる。これが BTB の精度が高い理由である。これに対し、一般に静的分岐予測が BTB に劣るのは、以下の理由による。

- ヒューリスティックによる予測と反対となる分岐命令があれば、その分岐命令に対する予測は実行中ほとんど外れてしまうことになる。たとえば、espresso のループの中には、loop exit jump がほとんど taken となるものが少なからずある。
- ヒューリスティックにあてはまらない分岐命令の割合が増えれば、ヒューリスティックによる効果は相対的に小さくなってくる。これは、espresso, xlisp, gcc 共にあてはまる。特に gcc では、ヒューリスティックにあてはまらない分岐命令の割合が多く、BTB との差が大きくなっている。

5 まとめ

静的分岐予測は、プログラムによっては非常に有効であり、BTB と同程度の精度が得られる場合がある。一般には BTB に及ばないが、BTB がない場合、あるいは BTB の容量が小さい場合に、Default のような単純な予測よりも効果があると考えられる。

6 今後の課題

我々の最終的な目標は、分岐予測を利用して性能を向上させることである。分岐予測の精度の向上が最終的にどのくらい性能向上に寄与するかは、ターゲットアーキテクチャ依存であり、この辺りの検討が今後の課題である。

参考文献

[1] T. Ball, J. R. LARUS: Branch Prediction For Free. ACM-SIGPLAN'93 Conference on Programming Language Design and Implementation, pp.300-313