動的最適化を前提としたホットパスの変化に関する予備評価*

5ZB−06

野中 雄一　小野 喬史　加藤 文彦　大津 金光　横田 隆史　馬場 敬信†
宇都宮大学工学部情報工学科‡

1　はじめに

プログラムは、コンパイル時に最適化を行うことにより、実行性能の向上が見込まれる。しかしながら、プログラムのループ部分において、頻繁に実行されるホットパスが、入力データによって変化したり、実行時に動的に変化する場合、コンパイラによる静的な最適化は困難となる。そこで、実行時に頻繁に実行されるホットパスの切り替わりを検出し、この情報から動的に最適化することにより、プログラムの性能向上が見込める可能性が出てくる[1]。

このホットパスを検出するプロファイリングにおいて、実行時に低オーバーヘッドで、かつどれだけ正確にプロファイリングができるかが問題となる。より正確にパスの情報を取れば、それだけ動的に最適化する効果が期待できる。

本研究では、パスの挙動を正確に検出することのできるEfficient Path Profiling[2]を用いて、プログラム実行時のホットパスの変化について調査を行ない、動的最適化の可能性について述べる。

2　Efficient Path Profiling

2.1　概要

Efficient Path Profiling(EPP)とは、プログラムのループ内における各基本ブロック間に、カウンタを挿入することで各パスに番号付けをし、どのパスが何回実行されたかを実行時に調べるプロファイリング手法である。

2.2　プログラムへの適用例

クイックソートプログラムを用いて、EPPを適用した際の例を示す。

図1は、クイックソートプログラムのループ部分のコントロールフローグラフである。各A〜Iは、ループ内における基本ブロックを示しており、C・Fはサブループとなっている。

図2は、このクイックソートプログラムにEPPを適用したものである。各基本ブロック間にある黒い四角はカウンタを表し、その横にあるr+=aは、そこでの処理を示している。これにより、例えばA→B→C→D→G→Hというパスが実行されたとすると、

$$r = 14 + 1 + 1 = 16$$

となり、最後にcount[16]がインクリメントされ、この16番目のパスが実行されたという情報が得られる。

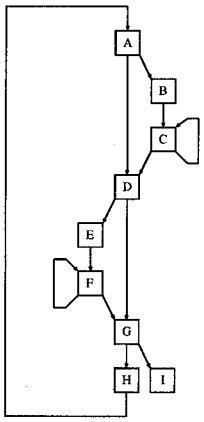

図1: quicksort

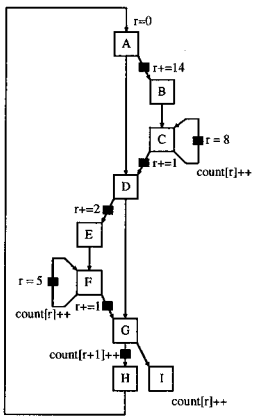

図2: EPP適用後

以下の表1において、各パスの番号付けを示す。

表1: Pathの番号付け

パス	番号	パス	番号
ADGI	0	CDGH	10
ADGH	1	CDEF	11
ADEF	2	CDEFGI	12
ADEFGI	3	CDEFGH	13
ADEFGH	4	ABC	14
F	5	ABCDGI	15
FGI	6	ABCDGH	16
FGH	7	ABCDEF	17
C	8	ABCDEFGI	18
CDGI	9	ABCDEFGH	19

3　実験

3.1　環境

実行のステージや入力データの違いなどにより、各パスがどのような挙動をしているか、またその情報からどのような最適化の可能性があるかを調べる。そこで、クイックソートプログラムのアセンブリコードに、EPPに

* Preliminary Evaluation of a Hot-Path Behavior
† Yuuichi Nonaka, Takahumi Ono, Fumihiko Katou, Kanemitsu Ootsu, Takashi Yokota, and Takanobu Baba
‡ Department of Information Science, Faculty of Engineering, Utsunomiya University

よるプロファイルのコードを挿入し、これに10000のランダムに並んだデータ、及び10000の昇順に並んだデータを入力した場合のパスプロファイリングを行う。

また、プロファイリングによる動的最適化を行う場合、プロファイリングのコストが、最適化による性能向上よりも上回ってしまうと、逆に全体の性能が下がってしまう。そこで、このプロファルのコードがどれだけのオーバーヘッドとなっているかを調べるため、EPPによるプロファイルのコードを挿入した時とそうでない時に、それぞれの実行サイクル数を計測し、これを求める。

3.2 実験結果

表1に示す0〜19のパスのうち、最適化の可能性があると考えられるものについて図3〜8に示す。横軸はパスの実行回数、縦軸は100回パスが実行された時、該当するパスがそのうち何回実行されていたかを表す。

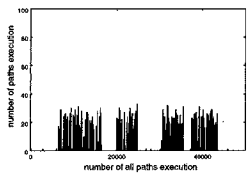

図 3: Path[0], ランダム

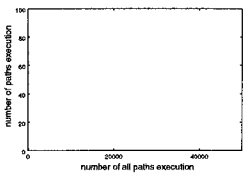

図 4: Path[0], 昇順

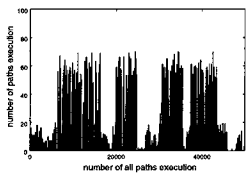

図 5: Path[1], ランダム

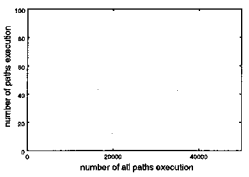

図 6: Path[1], 昇順

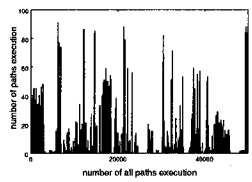

図 7: Path[5], ランダム

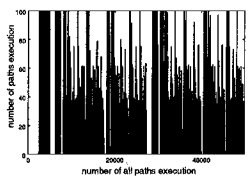

図 8: Path[5], 昇順

また、このときのEPPによるプロファイルのコードの部分のオーバーヘッドは、26.94%であった。

3.3 考察

図3と図5に着目すると、前者のパスの頻度が高いときには後者も、前者のパスの頻度が低いときには、後者のパスも低い傾向にある。この二つのパスとはADGIとADGHであり、共通部分であるADGについて最適化を行うことが有効ではないかと考えられる。

また、図3と図4、図5と図6から、入力データが違う場合に、パスの挙動もまったく違うものになることを示しており、このことからもこのパスについて動的最適化することによる速度向上の可能性がある。

図5及び図6のグラフでは、ある時は実行頻度が高く80%を越えているところもあり、ある時はほぼ0%となるなど、ホットパスの切り替わりが顕著に見られる。また、この二つの図から、入力データの違いによりパスの挙動が変わっていることも分かる。よって、このパスについても動的に最適化できる可能性があると考えられる。

4 おわりに

今回、パスがどのような挙動をしているかを実験で示したが、実行頻度がどれだけあったときに、それをホットパスとして決定し最適化の対象とするか、その閾値は定義していない。この閾値をどのように設定するかにより、最適化を切り替える回数やその効果が変わってくると思われ、これについても実験を行う必要がある。

また、EPPによるプロファイルのオーバーヘッドが動的最適化による性能向上よりも大きくなる可能性があり、これを少なくすることも視野にいれる必要がある。オーバーヘッドを削減する方法としては、プログラム実行開始から終了まですべてにおいてプロファイルを取るのではなく、サンプリング的に行う方法や、ホットパスの予測をする方法[3]が上げられる。しかしながら、オーバーヘッドを少なくしようとする場合、同時にプロファイリングの正確さも失われることとなり、ホットパスを誤検出する可能性も出てくる。そうなると最適化した際の速度向上も低くなる可能性がある。よって、このトレードオフについても調査する必要があると考えられる。

謝辞 本研究は、一部日本学術復興会科学研究費補助金(基盤研究(C)課題番号12680328)の援助による。

参考文献

[1] Michael D. Smith. “Overcoming the Challenges to Feedback-Directed Optimization,” Proc. ACM SIGPLAN Workshop on Dynamic and Adaptive Compilation and Optimization (Dynamo'00), invited lecture, Boston, MA, January 18, 2000.

[2] Thomas Ball, James R. Larus. “Efficient Path Profiling,” Proc. 29th Ann. IEEE/ACM Int'l Symp. Mycroarchitecture, IEEE CS Press, Los AAlamitos, Calif., pp. 46-57, 1996.

[3] Evelyn Duesterwald, Vasanth Bala. “Software Profiling for Hot Path Prediction: Less is More,” 9th ASPLOS , pp.202-211, 2000.