一時的プロファイルによるホットパス検出法の精度とオーバーヘッドの検討

加藤 文彦 野中 雄一 青木 政人 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

実行時にプログラムの挙動が変化する場合、コンパイラによる静的な最適化処理には限界がある。その場合、実行時にパスの挙動を調べてホットパスを検出し、その情報を基に動的に最適化を行うことで速度向上の可能性が出てくる。この時、正確でかつ低オーバーヘッドなプロファイリングを行うことが必要となる。しかし、正確なパスプロファイルを行う手法の中では比較的オーバーヘッドが小さいEfficient Path Profiling[1]を用いても、オーバーヘッドが100％程度になる場合があり、この両者にはトレードオフが存在する。

本研究では、プロファイルコードを含んだコードと含まないコードを状況に応じて切り替えることでオーバーヘッドを削減するプロファイリング手法“一時的プロファイル”を提案し、その手法の有用性について検討する。

2 パスプロファイル

2.1 一時的プロファイル

一時的プロファイルとは、Ephemeral Instrumentation [2]を他のプロファイル手法に応用したものである。

初めに、プログラムを関数単位に分割し、それぞれに対しプロファイル用のコードを挿入したコードと、元のコードの両方を用意する。パスをカウントした回数を測定し、それが閾値UTを超えた場合元のプログラムを実行する。これをunhookと定義する。そして、一定間隔RI毎に割り込みを掛けてパスカウントの総数をリセットし、再びプロファイルコード入りのコードを実行させる。これをrehookと定義する。unhookされてからrehookされるまでは元のコードを実行するため、プロファイルコードを実行する時間を減らすことができ、それによるオーバーヘッドの削減が期待できる。

2.2 Efficient Path Profiling

パスプロファイル手法であるEfficient Path Profiling(EPP)は、ソフトウェアのみでパスプロファイリングを行う手法のひとつであり、最小限の基本ブロック間にカウンタを挿入してパスの番号付けを行い、それによって、低オーバーヘッドで完全なプロファイリングを行うことができる。

その適用例を図1に示す。図の四角の部分はカウンタを挿入した部分を示し、隣に行った処理が記されている。rの式で書かれている部分を通る度に、パスの番号を示すカウンタrを加算し、バックエッジもしくは関数の呼び戻し部分でrで示される番号のパスの実行回数のカウントを行う。

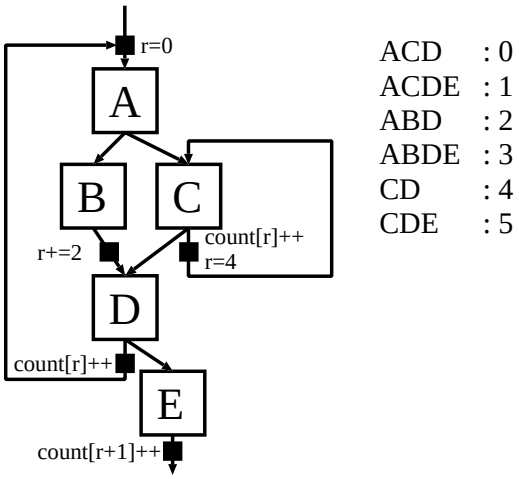

図 1. EPPの適用例

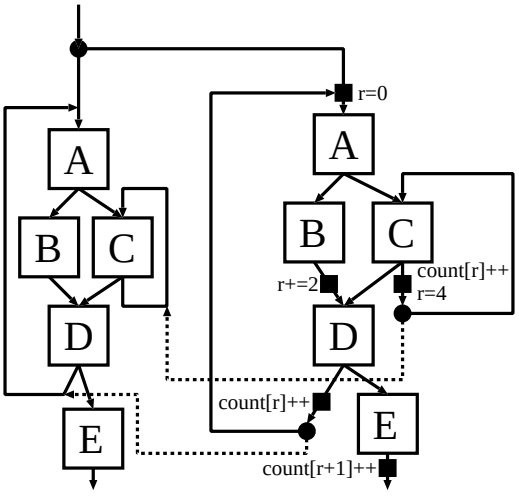

図 2. 一時的プロファイルのEPPへの適用例

2.3 一時的プロファイル法によるEPP

EPPを用いたプロファイルコードに対して一時的プロファイルの適用した例を、図2に示す。一時的プロファイルによって追加されたコードは、図中の黒丸の

A consideration of the accuracy and overhead of hot path detection by temporary profiling
† Fumihiko Katou, Yuuichi Nonaka, Masato Aoki, Kanemitsu Ootsu, Takashi Yokota and Takanobu Baba
‡ Department of Information Science, Faculty of Engineering, Utsunomiya University

部分である。

初めに、関数の先頭部分で関数内のパスのカウント数の合計がUT以下の場合EPPを適用したコードに分岐し、そうでなければ元のコードを実行する。そして、EPPを適用したコードの中のバックエッジ部分に、パスをカウントした回数の合計をカウントし、それがUTに達したときは、図の破線で示されている元のコードの対応する部分へと分岐を行う。

3 評価

3.1 評価方法

一時的プロファイルを用いることで、EPPと比べオーバーヘッドやパスの検出結果がどう変化するかを調べるため、以下の条件で評価を行った。評価プログラムには、要素数1000の乱数(0〜999)を昇順にソートするクイックソートのプログラムを用い、一時的プロファイルの再びプロファイルを取り始める間隔RIを1000000クロックとし、プロファイルを取るのを止める閾値UTを50, 100, 500, 1000, 5000, 10000と変化させた。

3.2 評価結果

閾値UTごとのオーバーヘッドは図3のようになった。ここで言うオーバーヘッドは次式のように定義する。

$$\text{オーバーヘッド}=\frac{\text{プロファイルを取るプログラムの実行サイクル数}-\text{元のプログラムの実行サイクル数}}{\text{元のプログラムの実行サイクル数}}\times 100(\%)$$

EPPでは26.5％ものオーバーヘッドが発生するのに対し、一時的プロファイルでは、2.6％〜52.8％となった。

このことから、閾値を低くすることで、オーバーヘッドを低く抑えられることが分かる。しかし、閾値を大きくしていくと、一時的プロファイルではプロファイルを完全に取っていないにもかかわらず、プロファイルを完全に取るEPPよりもオーバーヘッドが大きくなってしまう。

これは、一時的プロファイルではEPPと比べて、EPPを適用したプログラムと元のプログラムを切り替えるコードや、関数内のパスカウント数の総数をカウントするコードが増えているためであると考えられる。

次に、実行頻度は次式のように定義し、EPPにおいて実行頻度が高かった6種類のパスについて、閾値を変化させたときの実行頻度を図4に示す。

$$\text{実行頻度}=\frac{\text{パスのカウント数}}{\text{全パスのカウント数の総数}}\times 100(\%)$$

図は各パスについて左から、EPP, 一時的プロファイルの閾値50, 100, 500, 1000, 5000, 10000の実行頻度となっている。

この図から、閾値が50や100の場合、半数のパスが検出できていないことが分かる。しかし、閾値が500, 1000のときはPath57の実行頻度が、EPPでは12.9％であるのに対し、それぞれ1.6％, 1.9％と低くなっているものの、そのパスが比較的頻繁に実行されていることは分かるため、大体の傾向はつかめていると言える。

以上のことから、一時的プロファイルは、閾値を適切な値に設定することである程度の精度を保ったまま、オーバーヘッドの削減が可能であると言える。

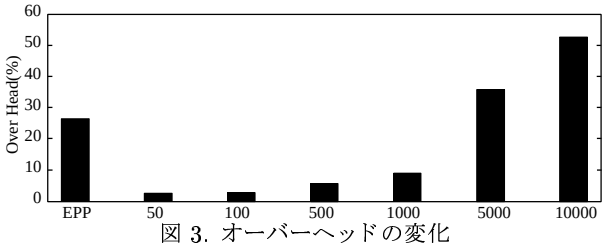

図 3. オーバーヘッドの変化

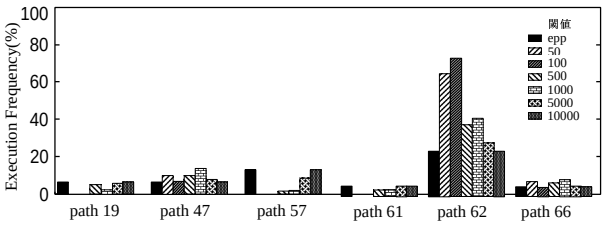

図 4. ホットパス検出精度の変化

4 おわりに

本稿では、プロファイルコードの付加されたプログラムと元のプログラムを実行時に切り替える一時的プロファイルを提案し、それを既存のパスプロファイルアルゴリズムであるEPPに適用することで、オーバーヘッドを大幅に削減できる事を示した。

今後の課題として、様々なプログラムに対して提案手法を試し、閾値や割り込みの間隔をどのように設定すると有効か検討することが挙げられる。また、本手法では、関数の中を実行している時にrehookが行われた場合、その関数から一度戻って再び呼び出すまでプロファイルが行われないため、一度呼び出すと長い時間実行するような関数では、プロファイルがほとんど行われないといった問題があり、提案手法のアルゴリズムの改良が必要であると考えられる。

謝辞 本研究は，一部日本学術振興会科学研究費補助金(基盤研究(B)14380135, 同(C)14580362, 若手研究14780186)の援助による.

参考文献

[1] Thomas Ball, and James Larus, “Efficient Path Profiling,” Proc. of the 29th annual IEEE /ACM international symposium on Microarchitecture (MICRO-29), pp.46-57, December 1996

[2] Omri Traub, Stuart Schechter, and Michael D. Smith, “Ephemeral Instrumentation for Lightweight Program Profiling,” http://www.eecs.harvard.edu/hube/publications/muck.pdf, June 2000.