A-028

Squeakにおけるメソッド配分プロファイラの実装を目的としたVM生成システムの拡張

Extending VM generation system for impremention of method dispatch profiler in Squeak

長田 忍† 楢崎 修二‡
Shinobu Nagata Shuji Narazaki

1 はじめに

オブジェクト指向プログラミング言語 (以下 OOPL) はオブジェクトに対するメッセージ送信によって処理を実現している。受け取ったメッセージを処理すべき実体がどのクラスのどのメソッドであるかを探索することをメソッド配分と言い、メッセージ送信の度に探索を行うので実行時における計算量の増加を避けることができず処理時間も増えてしまう。これを改善するためメッセージ送信の高速化について様々な手法 [4] が提案されている。例えばオブジェクトに送られたメッセージの名前 (セレクタ) とメッセージが送られたオブジェクト (レシーバ) のクラスとの組を仮想マシン (VM) が持つキャッシュに登録しておくキャッシュ法や、メソッド配分の統計データを用いてレシーバのクラスを比較することで探索を省くレシーバクラス予測 [1] といった手法がある。またメッセージ送信の高速化に利用する統計データを取得するためにはメソッド配分のプロファイリングを行う必要があり [3]、メソッド配分のプロファイリングを行ってボトルネックを探し出し改善することができれば実行時間の短縮が望める。

本研究はメッセージ送信を高速化するための情報収集機構として、OOPL の一つである Squeak の VM にメソッド配分のプロファイラを実装することを目的としている。

2 Squeakにおけるプロファイリングの問題

OOPL におけるプロファイラはメソッドの呼び出し回数、処理にかかった時間などをプロファイリングするものが多い。プロファイリングはプログラムの解析を行うためのものであり、この点で OOPL にとって必要な機能であると言える。

Java や Python などの OOPL はプロファイラを持っており、Squeak はプロセスを監視する ProcessBrowser によって Squeak 上でメッセージ送信のプロファイリングを行っている。しかし ProcessBrowser は Squeak 内の指定したプロセス上において指定した時間のみプロファイリングを行うものであり、全てのメッセージ送信についてプロファイリングできない。また Squeak はプログラミング言語であると同時に様々な作業を行う統合環境でもあり、Squeak 上での処理全てがメッセージ送信によるものであることから、統合環境としての Squeak におけるメソッド配分のボトルネックを探すためには Squeak 上でなく VM 上にプロファイラを実装するのが望ましい。

以上を踏まえて VM 上にメソッド配分のプロファイラを実装することとした。Squeak は VMMaker という VM 生成システムを持っており、今回メソッド配分のプロファイラを実装する手段として VMMaker の拡張を行う。また Squeak VM はプリミティブという命令を持っており、ユーザが定義できるプリミティブを使うことで Squeak 上から VM を操作することができる。今回 VM にプロファイラを実装する際にプリミティブを使って Squeak 上でプロファイリング機能の有効化と無効化とを行えるように設計した。通常は変更後の VM 上でプロファイリングを行い起動から終了までのデータを出力するが、ある作業を行う間だけプロファイリングを行うことも可能となる。

3 実装

3.1 メソッドキャッシュの利用

Squeak でのメッセージ送信はセレクタと引数とを組にしたメッセージをオブジェクトへ送信することである。Java では同じセレクタのメソッドが複数存在する場合もあるが、Squeak では一つのセレクタは一つのメソッドのみが使用するため引数を考慮せずセレクタとクラスのみを用いてメソッド配分を行う。また Squeak はメソッ

†長崎大学 大学院 生産科学研究科
‡長崎大学 工学部

ド配分にキャッシュ法 [4] を利用しており、探索を行う前にレシーバとセレクタを使ってハッシュ値を計算しメソッドキャッシュを検索する。キャッシュにヒットしなかった場合は、まずレシーバのクラスを調べてから、

1. クラスが理解できるメッセージのリスト (そのクラスのメソッド辞書) からメソッドを探し、
2. メソッドが見つからなければ現在探索を行っているクラスのスーパークラスへと探索対象を変える、

という処理を繰り返して探索を行い、その結果を新しくキャッシュに登録する [2]。

このメソッドキャッシュ上の一つ一つのエントリにプロファイリング用のフィールドを用意し、キャッシュから追い出されたりキャッシュのフラッシュが起きた時にその時点での配分回数を新たに作ったメソッド配分のカウント表へ書き戻す、という処理を行うことでプロファイリングを行う。

3.2 Squeak の VM 生成システムの拡張

Squeak の VM は VMMaker というツールを使って生成する。具体的には

1. まずユーザが Slang で VM のモデルとなる `Interpreter` クラスを記述し、
2. 次に `CCodeGenerator` クラスがモデルを C 言語に変換して VM のソースコードを生成する、

という手順で VM を生成する。ここで Slang とは Squeak 自身が解釈して実行するプログラミング言語 Smalltalk を C 言語に変換することを意識しながら記述する Smalltalk と C 言語とのハイブリッドなプログラミング言語である [2]。

今回、図 1 のように `Interpreter` クラスの該当箇所にプロファイリングのためのコードを挿入した新しいクラスを `Interpreter` クラスと追加メソッド用のクラスから自動で作り出すようにシステムを拡張した。この拡張は、VMMaker がプロファイリング情報を使って最適化を施したコードを織り込んだ新しい VM を自動的に作成することができるような改良を視野に入れたものである。

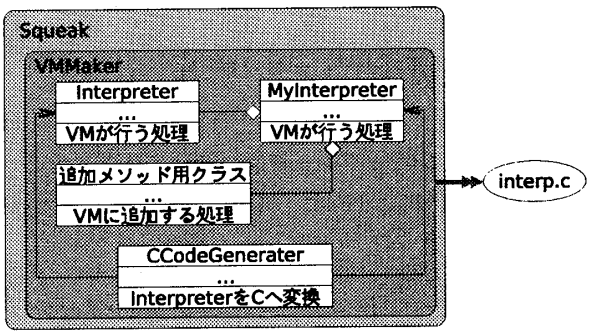

図 1: 拡張した VMMaker

4 まとめ

本研究では OOPL である Squeak におけるメッセージ送信を高速化するための情報収集手段としてメソッド配分のプロファイリングに注目し、その機能を Squeak VM に実装した。Squeak は自身のライブラリに定義してある `Interpreter` クラスをモデルとして C 言語に変換された VM を生成する仕組みを持つ。今回そのシステムの拡張を行い、メソッド配分のプロファイリング処理を記述した自作クラスと `Interpreter` クラスとの結合を行うことでメソッド配分のプロファイラを VM へ容易に実装できる環境を整えた。

今回メソッド配分のプロファイラを実装した VM と通常の VM とでベンチマークプログラムを実行したところ、プロファイラを実装した VM 上では通常の VM より実行時間が 1.6 倍程度長くなった。これはプロファイリング結果として出力するために行う、クラス名とセレクタ名とを取得する処理で生まれたオーバーヘッドによるものであると考えられる。また実装したプロファイラはメソッドが配分された回数についてのみ調べており、メソッドの処理にかかった時間については調べていない。以上から今後の課題としてプロファイリングによるオーバーヘッドと得られる情報量とのトレードオフについての評価を挙げておく。

参考文献

[1] David Grove, Jeffrey Dean, Charles Garrett, and Craig Chambers. Profile-Guided Receiver Class Prediction. ACM SIGPLAN Notices, Vol. 30, No. 10, pp. 108–123, Oct. 1995.

[2] Mark J. Guzdial and Kimberly M. Rose. Squeak Open Personal Computing and Multimedia. Pearson Education, 2002.

[3] 神尾貴博, 増原英彦. オブジェクト指向プログラムの高速化を支援するプロファイラ. 情報処理学会論文誌 (プログラミング), Vol. 46, No. SIG1, pp. 1–9, Jan. 2005.

[4] 小野寺民也. オブジェクト指向言語におけるメッセージ送信の高速化技法. 情報処理学会誌, Vol. 38, No. 4, pp. 301–310, Apr. 1997.