デ-08

SH マイコン向け JIT コンパイラの実装

○井奥 章† 北沢 俊太郎‡ 一瀬 成広‡ 川崎 進一郎† 森本 義章† 磯部 竜雄‡

(株)日立製作所 システム開発研究所† 日立ソフトウェアエンジニアリング（株）‡

1 背景：Java™ の適用拡大

Java™ はバイトコードと呼ばれる機械非依存な中間命令を解釈実行するインタプリタ実行系であるため、異なるハードウェア間で同じプログラムを動かすことができる特徴を有している。そのため、組込み機器で Java™ を利用すると次のような利点が生まれる。

（1）高いソフト生産性：PC 上のツールが利用可能な上、開発したプログラムが実機で即座に動かせる。

（2）組込み機器上のサービス拡大：ハード提供メーカ以外のベンダによるプログラム開発が可能になる。また、異機種間でのデータ交換や通信の標準化が進んでおり様々な新サービスが期待できる。

このような利点が注目されて、Java™ は,携帯電話などの小型の情報機器から制御機器に至るまで搭載事例が増えている。

たとえば、制御システム分野では、IP ネットワークを活用した新サービスの提供・運用・保守の効率化が求められている。そのため、システム構築の核となるコントローラなどでは、インターネット対応・プラットフォーム独立・管理コスト低減の狙いから、Java™ 実行環境のサポートが重要となってきている。

2 組込み機器向けの技術課題

組込み機器の多機能化にともない、コストパフォーマンスの要求水準がさらに高まっている。そこで、インタプリタ方式であることに起因する Java™ の実行性能上の弱点が、課題視される場合がある。組込み機器で Java™ を利用する場合を考えると、特に、以下のような技術課題が存在する。

（1）C 言語やアセンブラに比べ実行速度が遅い
（2）リアルタイム性が期待できない
（3）Java 実行環境自体のサイズが大きい
（4）利用するメモリ量の予測が困難

本稿ではインタプリタ部を最適化する技術として知られる、JIT(Just-in-Time)コンパイラ技術の実装事例を採り上げ、課題への効果、主に（1）の改善について、議論する。JIT コンパイラ技術は、バイトコードを実行時に機械語命令へ変換する技術である。

Implementation of JIT Compiler for SuperH Risc Engine
Akira Ioku†, Shuntaro Kitazawa‡, Naruhiro Ichise‡,
Shin-ichiro Kawasaki†, Yoshiaki Morimoto†, Tatsuo Isobe‡
†Hitachi,Ltd., Systems Development Laboratory
‡Hitachi Software Engineering Co., Ltd., 2nd Software Development Division

3 J2ME/CDC の Dynamic Compiler の概要

各種組込み機器のニーズへの即応体制を強化するため、Sun Microsystems 社が、最新の組込み Java™ 仕様に沿った Java™ 実行環境 J2ME/CDC[1]を提供している。J2ME/CDC は広範な組込み機器に対応するため、プロファイルによって Java 実行環境の機能を取捨選択できる。

J2ME/CDC HI(Hotspot Implementation) [2]は、最新の「Dynamic Compiler」という名称の JIT コンパイラ搭載の組込み機器向け Java™ 実行環境である。Dynamic Compiler は中間表現を介在させる 2 パス構成を有する JIT コンパイラである。中間表現を介在させるために、生成コードの高度な最適化が可能である。

4 SH マイコン向けの対応

筆者らは組込み機器向けマイコンである（株）ルネサステクノロジ製 SuperH™ RISC engine[3]（以下「SH マイコン」）に特化した Java™ 実行環境の性能改善を進めてきた。さまざまな組込み機器に SH マイコンは広く採用されているマイコンである。

以下に、J2ME/CDC HI の Dynamic Compiler を SH マイコンに対応させて、一層の高速化を実現した成果を、報告する。

4．1 SH マイコン仕様への適合、最適化

（1）キャッシュのコントロール

JIT コンパイラでは、バイトコードから変換したマシンコードをメモリ上のデータエリアに格納し、それをプログラムコードとみなして強制的に実行するのが通例である。今回対応させた SH-3 では、データキャッシュと命令キャッシュが区別されないため、データが命令とみなされても問題がない。

ところが、今回対応させた SH-4 の場合、データキャッシュと命令キャッシュは別々である。このため、変換したマシンコードは、データとして処理され、データキャッシュ上にコピーされる。もし、そのまま、同じアドレスを実行すると命令キャッシュ上には変換されたマシンコードとは異なる意図しない値が残っており、うまく動作しない。このため、SH-4 版では、コンパイル直後にキャッシュをフラッシュする必要がある。

（2）適切な CPU 命令の選択

JIT コンパイラを実現する際には、CPU の仕様への適合が安定的な動作に必要であるとともに、その効果的な活用が、実行性能面での最適化に有効である。

たとえば、一般のコンパイラの最適化手法で知られているように、マイコン命令のなかで、有利な命令を選択することは、SH マイコン向けの JIT コンパイラへの適用においても重要であった。一例としては浮動小数点演算

のカスタマイズが挙げられる。

SH-4 では、32 ビット短精度浮動小数点演算と 64 ビット浮動小数点演算命令が用意されている。しかし、このような専用命令を有しないマイコン（SH-3）では、汎用レジスタなどのソフトウェア処理で代替した演算ライブラリを呼び出すことで浮動小数点演算を実現せざるをえない。

SH-4 では、この命令を使うようにする、また、整数の割り算も、浮動小数に変換してから割り算を行った後に整数へ変換する等、これらの処理を SH-4 の命令として用意できれば、ライブラリを呼び出すより高速に実行できる。

その他、同期制御において、SH マイコンの高速な同期命令（Test And Set(tas.b)命令）を積極的に利用するなど、適切な CPU 命令の配剤が高速化に寄与する。

（3）メモリのレイアウト

SH マイコンは、ポストインクリメント・アドレッシングによるロード命令、およびプリデクリメント・アドレッシングによるストア命令を持っている。このため、0 アドレスに向かって伸長するスタックに対するアクセスは 1 命令で実行できる。しかし、ポストインクリメント・アドレッシングによるストア命令、およびプレデクリメント・アドレッシングによるロード命令が無いため、プラス方向に伸長するスタックへのアクセスには 2 命令かかることになる。

また、SH マイコンは、正の定数オフセットによる配列形式データアクセスに適したアドレッシングモードを備えている。しかし、負の定数オフセットに対応したアドレッシングモードは持っていない。このため、配列アクセス時には、基準アドレスから正方向に伸びるデータ構造が適している。

スタックマシンである Java™ のインタプリタは、オペランドスタックへのアクセスが頻発する。したがって、SH マイコンを用いた場合、スタックの伸長方向次第で実行効率が大きく異なる。この点を意識したメモリレイアウトで設計することが望ましい。

4．2　製品化

本研究開発の成果は、日立ソフトウェアエンジニアリング（株）製 SuperJ Engine® V3．2として製品化されている。

SuperJ Engine®シリーズ[4]は、SH マイコン用の高性能な Java™ 実行環境であり、以下の適用事例が存在する。

- エレベータ操作パネル
- 下水システム監視用サーバ
- 業務用車載システム
- 環境監視用サーバ

V3．2は、J2ME/CDC HI に準拠した製品である。4.1 に示した施策の一部を実装した状態で、完成度を高め、SunMicrosystems 社の Java™ 互換性テストプログラムにパスし、Java™ 互換認証を受けた製品であり、高性能ながら高い動作安定性を実現している。

4．3　評価

本評価では、SuperJ Engine® V3．2の性能を計測した。性能を計測するベンチマークプログラムは、組込み分野で広く用いられている Embedded Caffeinemark3.0 を用いた。OS は、SH-Linux である。

Embedded CaffeineMark3.0 各計測項目の結果を図 1 に示す。値が大きいほど高速である。下段が SuperJ Engine® V3．2であり、上段が、Dynamic Compiler を適用していないこと以外は同等の実装品である。10 倍程度の高速化を達成した結果を示しており、十分な性能向上効果を確認できる。

5　おわりに～JIT コンパイラの課題への対応

JIT コンパイラが生成する機械語は、バイトコードに比べ平均して一桁ほどサイズが大きいといわれている。メモリを節約するにはインタプリタで実行する方が得策ともいわれている。また、コンパイル時間のオーバーヘッドを問題視する向きもある。

Dynamic Compiler では、これらの懸念事項に対して次のように対処している。

（1）選択的なコンパイルを可能にしている

（2）計算機資源の消費量やコンパイル条件のチューニングを可能にしている

（3）コンパイル時間、資源消費を過剰に要求する最適化を避ける

結果として、本システムにおいては、メモリ消費量の増大の抑制を確認できている。この点についても、デモンストレーションでは言及する。

6　参考文献

[1]http://java.sun.com/products/cdc/
[2]http://java.sun.com/products/cdc-hi/index.jsp
[3]http://www.renesas.com/jpn/products/mpumcu/32bit/sh/
[4]http://www.hitachi-sk.co.jp/Products/SuperJ/

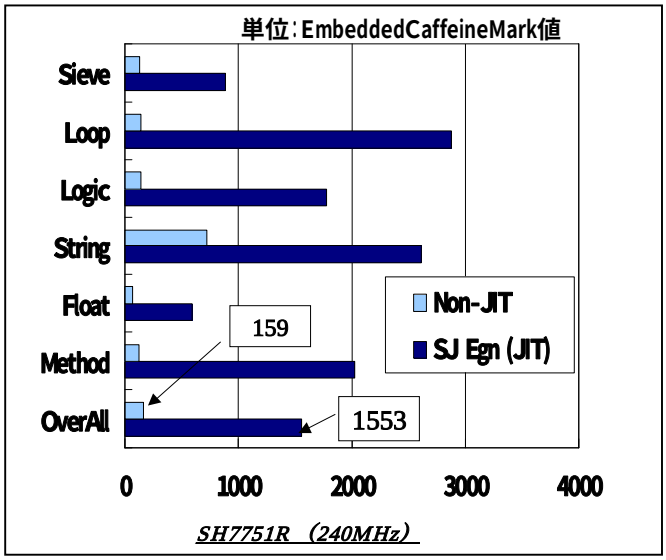

図1　SuperJ Engine® V3．2(SJEgn)の性能