JavaVM上で動作するx86ユーザーモードエミュレータの実装と評価

川口　直也†　並木　美太郎‡

東京農工大学大学院工学府情報工学専攻†　東京農工大学大学院共生科学技術研究院‡

1.はじめに

ネイティブコードからなるユーザーモードアプリケーションの異なるOS、アーキテクチャ間での可搬的な利用が困難な理由として、コンパイル済みバイナリの機械語命令列とOSのシステムコールが挙げられる。

このうちシステムコールについては、機械語命令に互換性のあるアーキテクチャ上であっても、OSが提供する機能に大きく依存するため、ファイルシステムなどの周辺機能も含めてハードウェア機構全体を仮想化するフルシステムエミュレーションによって解決する方法が試みられている[1]。しかし、アプリケーションからは直接利用することのない特権命令や入出力デバイスを含めてエミュレーションするこの方式では、単にユーザーモードの環境を模倣するという観点からは、エミュレーションに要する計算機資源が過大である。

他方、ユーザーモードにおける機械語命令のエミュレーションのみを行い、システムコールは、その環境のネイティブなOSに対し処理を依頼するユーザーモードエミュレーション[2][3][4][5][6]は、エミュレータの動作環境に存在するOS資源をアプリケーションから直接利用出来るという点において有用である。しかし、この方式では動作環境が同一のシステムコールライブラリを実装したOS間に限定されてしまう。

本報告では、ユーザーモードエミュレーションにおけるこの問題を解決し、異種OS間におけるx86/Linuxアプリケーションの可搬的利用を目的とした、JavaVM上で動作するユーザーモードエミューレータとその性能評価を示す。

JavaVM上で動作する本エミュレータは、エミュレータ自身が可搬性を持ち、またLinux以外のPOSIX-APIを提供するOS上において動作するよう非POSIX準拠のシステムコールのエミュレーションを行う。

2.目標

筆者は、今日一般的に使われているLinuxアプリケーションが本エミュレータの上で動作することを期した。具体的には、I/O処理やユーザからの入力待ちにおいて大部分の時間を消費し、CPU命令のエミュレーションによる性能低下の影響が限定的と考えられる、GUIプログラムやLinux用の標準コマンドがこれに該当する。

また Linux 用の GUI プログラムが Java アプレットとして Web ブラウザ上で動作することも期した。このため、本エミュレータの実装にあたり、以下の点を目標とした。

- x86/Linux向けのELFバイナリを実行可能
- 共有ライブラリをロード可能
- Linux以外のPOSIX互換OS上で動作する
- 可能な限りJavaで記述しエミュレータ自身にも可搬性を持たせる
- GUIプログラムが動作する
- マルチスレッドプログラムが動作する
- Javaアプレット化が可能である

3. 設計

図1に本エミュレータの構成を示す。x86/Linux用のユーザーモードプログラムを構成する機械語命令列のうち、内部割込み以外の非特権命令は本エミュレータ上でエミュレーションする。システムコールを利用するための内部割込みはJavaVMの下で動作するJNI(Java Native Interface)モジュールを経由して、実機のOSが提供するシステムコール資源を利用する。これにより、ネットワークやファイル I/O といったハードウェア資源を仮想化することなくアプリケーションから直接利用可能とする。

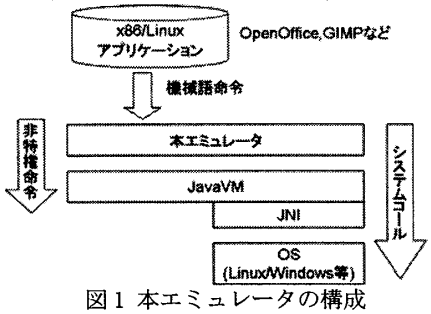

図1 本エミュレータの構成

4.評価

本エミュレータでは ls, wc コマンド等の簡易なプログラムのほか、マルチスレッドや共有ライブラリのロードをサポートしている。さらに xlib をリンクした簡単なGUIプログラムの動作も確認している。

以下に本エミュレータの基本性能と、Linux コマンドの実行時性能、Linux 用の GUI プログラムをアプレット化した例を示す。

基本性能

本エミュレータ上での性能が、実機上でプログラムが動作した場合に比べ、どの程度低速になるかを示す。対象としたプログラムの処理内容はフィボナッチ、空の for ループ、ADD 命令を線形に配置した分岐の無いコード、及び/dev/zero に対する read、/dev/null に対する write

---

Development and evaluation of an x86 user-mode emulator on JavaVM

† Naoya Kawaguchi
Computer and Information Sciences, Tokyo University of Agriculture and Technology
‡ Mitaro Namiki
Institute of Symbiotic Science and Technology, Tokyo University of Agriculture and Technology

である。

表 1 に本エミュレータ上での各プログラムの実行時性能が、実機で実行した場合に比べ何倍低下したかを示す。

エミュレータ上での分岐の無いコードの実行時性能が特に低下する理由として、本エミュレータでは命令デコード結果をキャッシュして再使用することが挙げられる。

また、read/write システムコールの結果は、本エミュレータの有用性が純粋な演算処理よりも、I/O 処理において発揮されることを示している。

表 1　本エミュレータと実機との基本性能比率

プログラム	実機に対する性能低下比率
フィボナッチ	2 万～7 万
空の for ループ	10 万～14 万
分岐の無いコード	14 万～42 万
read(/dev/zero)	2000～3000
write(/dev/null)	1600～3 万

Linux用コマンドの性能

Linux に標準で添付されている 4 つのコマンドプログラムの実行時性能を測定した。本エミュレータ上での実行に要した時間を実機上での実行時間で割ったものを表 2 に示す。ls は 15 個のファイルをオプションなしで表示した。wc は 112 個のファイルに分散した Java 言語のソースファイル 6 万行程度の行数を取得した。gzip は約 7kb のファイルを圧縮した。dd は/dev/zero から /dev/null に 20MB のデータを読み書きした。なお、これらのプログラムは libc.so 等の共有ライブラリを実行時に動的にリンクするものである。

大幅な性能低下が見られるが、これらの実際的なプログラムではシステムコールを多用するため、先に示した基本性能と比べ良い性能を示した。

表 2. Linux コマンドの実行時間比率

コマンド	実機に対する性能低下比率
ls	1 万
wc	5.5 万
gzip	1.2 万
dd	7500

GUIとエミュレータのアプレット化

本エミュレータと Java 言語で書かれた X サーバである weirdX を組み合わせることにより、x86/Linux 用の GUI プログラムが Java アプレットとして Web ブラウザ上で動作することを可能とした。以下では x86/Linux 用の GUI プログラムを Windows 上で実行したものを示す。

図 2 にはウィンドウを表示するだけの簡易な GUI プログラムの動作画面を示し、図 3,4 にはこれをアプレット化したものを示す。ブラウザの画面内でウィンドウを描画した場合、ウィンドウマネージャの不備により、ウィンドウの拡縮といった基本操作は実現できていない。一方、ブラウザの画面外でウィンドウを描画した場合、動作環境のウィンドウ管理システムと連携するため、マルチウィンドウのプログラム等も支障なく操作可能である。

図 5 には簡易なペイント機能を提供する GUI プログラムを、同じくアプレットとして実行したものを示す。実機上ではユーザからのマウス入力に対し、ほぼ遅滞なく画面が更新されるのに対し、本エミュレータ上では画面が更新されるまでに約 250 秒を要している。このため、本エミュレータは既存の GUI プログラムをアプレット化し、Web ブラウザ上に配信することを可能としているが、対話的操作に必要な性能をまだ有していない。

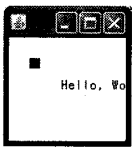

図 2 元の GUI プログラム

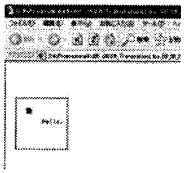
図 3 アプレット化（ブラウザ内描画）

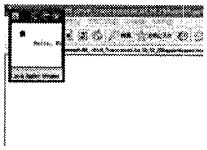
図 4 アプレット化（ブラウザ外描画）


図 5 アプレット化した GUI プログラム

5.終わりに

x86/Linux 用プログラムを JavaVM 上で実行するためのユーザーモードエミュレータを実装し評価を行った。本エミュレータは x86 命令セットのうち、浮動小数演算命令以外の全ての非特権命令と、FPU 命令の一部を実装している。また、非 POSIX システムコールのエミュレーションとして、clone システムコールを pthread_create で代替するといった機能を有している。結果、大幅な性能低下が見られたものの、GUI や共有ライブラリの動的リンク、マルチスレッドプログラムのサポートを実現している。

今後の展開として性能の向上と、より多くの非 POSIX 互換システムコールをサポートすることによって、OpenOffice 等の大規模なプログラムをエミュレーションすることが考えられる。また、NestedVM[3] のように POSIX 互換システムコールを Java 言語の標準ライブラリでエミュレーションすることで、JNI を使用しない pure-Java なエミュレーションの実現が考えられる。

参考文献

1) Rhys Newman,et al, “JPC:ピュア Java の x86PC エミュレータ”,Beautiful Architecture, pp 205-239, オライリージャパン,2009.
2) QEMU Project, http://bellard.org/qemu/
3) ALLIET, B., AND MEGACZ, A. Complete translation of unsafe native code to safe bytecode. *In Proceedings of the 2004 Workshop on Interpreters, Virtual Machines and Emulators*, pp. 32–41,2004.
4) D.Ung, C.Cifuentes. “Machine-adaptable dynamic binary translation”, *proc. of the ACM SIGPLAN workshop on Dynamic and adaptive compilation and optimization*, pp.41-51,2000.
5) R.Sites et al., ”Binary Translation”, *Communications of the ACM volume 36*, pp69-81, 1993
6) Linux/Alpha project, “EM86: How To Run Linux/x86 Apps on Linux/Alpha”, http://www.alphalinux.org/faq/FAQ-16.html
7) Intel, “IA32 Intel アーキテクチャ・ソフトウェア・デベロッパーズ・マニュアル”, Intel Corporation, 2004