複合ハイパバイザによる仮想計算機の高速化

榮樂英樹† 新城 靖†† 加藤和彦††

1. はじめに

ユーザレベル OS とは、実機上で動作する OS を、他の OS 上でユーザプロセスとして動作可能にしたものである。これまでに、VM とユーザレベル OS の両方の技術を用いるユーザレベル OS の実現方式を提案し、ハードウェアのエミュレーションを行う VM である LilyVM と、機械語命令の静的な変換を行うアセンブラのプリプロセッサを実現した[1)]。しかし、従来の LilyVM の実現では、ユーザレベルのハイパバイザだけを用いていたため、以下のようなオーバヘッドが大きかった。

- フォールトおよびトラップのエミュレーションのオーバヘッド
- MMU のエミュレーションのオーバヘッド
- FPU レジスタ切り替えのオーバヘッド

本研究では、ホスト OS にカーネル・モジュールを組み込むことによってハイパバイザを追加し、これらのオーバヘッドを削減する。フォールトおよびトラップは、ハイパバイザでエミュレーションを行うことでコンテキスト切り替えを不要とする。MMU のエミュレーションは、ハイパバイザでページ・テーブルを操作することで、オーバヘッドを削減する。FPU レジスタの切り替えは、ハイパバイザコールを用いて、FPU レジスタを使用している場合にのみ行われるようにする。さらに、このようなカーネル・モジュールを、複数の OS に簡単に移植できるようにする。

2. 設計と実現

2.1 基本設計

本研究で提案する仮想計算機の構成を図 1 に示す。ゲスト OS は、以下のプログラム群で作られた仮想環境の中で動作している（図 1）。

- カーネルハイパバイザ（Kernel Hypervisor）：カーネルレベルで動作するハイパバイザ
- VM プロセス（VM process）：ユーザレベルで動作するハイパバイザ
- VM ライブラリ（VM library）
- カーネル・モジュール（Kernel Module）

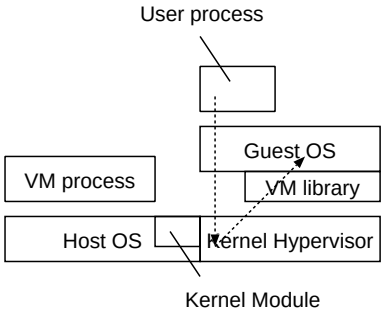

図 1 本研究で提案する仮想計算機の構成

ゲスト OS のカーネルは、LilyVM で用いているものと同様に、機械語命令の静的な変換を行ったものである。機械語命令のエミュレーションは、ゲスト OS と同じメモリ空間にある VM ライブラリの手続きによって行われる。

図 1 の点線は、ゲスト OS の実行中に発生したフォールトまたはトラップをエミュレートする時の制御の流れを表している。従来の LilyVM では、ホスト OS の機能を用いてフォールトおよびトラップを検出していたため、1 回のエミュレーションにつき 2 回のコンテキスト切り替えが必要であった。これを、図 1 に示すように、カーネルハイパバイザがリダイレクションを行うことで、コンテキスト切り替えが起こらないようにする。なお、FPU レジスタに関するフォールトと、ページ・フォールトについては、別の処理を行っている。詳しくはそれぞれ 2.2 節、2.3 節で述べる。

VM プロセスは、ホスト OS のシステム・コールを発行し入出力を行うユーザ・プロセスである。VM プロセスには、仮想計算環境のためのメモリを確保する役割もある。VM プロセスがカーネル・モジュールに対してゲスト OS の実行を命令すると、ハードウェア割り込みが発生するか、停止命令が実行されるか、

† 筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻
†† 筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻／ JST CREST

ディスクドライブ、ネットワーク等への入出力要求があるまでの間、ゲスト OS が実行される。VM プロセスは、ゲスト OS の実行が停止すると、ハードウェア割り込みのエミュレーションを行うか、入出力の要求を処理する。このような構成にすることによって、ホスト OS から見ると、ゲスト OS が実行されていることは完全に隠されており、単に VM プロセスが呼び出したカーネル・モジュールが実行されているようにしか見えない。

カーネル・モジュールは、システム・コール等を通じて VM プロセスと通信する。さらに、割り込みベクタやセグメント等を切り替え、カーネルハイパバイザにコンテキストを切り替える機能を持つ。カーネル・モジュールは、ゲスト OS を実行する時、カーネルハイパバイザにコンテキスト切り替えを行う。実行がカーネルハイパバイザから戻ると、カーネル・モジュールは、ゲスト OS が停止した原因を調べる。もし、ハードウェア割り込みが原因であった場合は、本来のホスト OS の割り込み処理ルーチンを呼び出す。

2.2 FPU レジスタ

LilyVM では、FPU を使わない場合でもコンテキスト切り替えで FPU レジスタの内容を入れ替える必要があった。その理由は、IA-32 の持っている遅延評価機能がユーザ・プロセスから利用できないためである。

カーネルハイパバイザを使用すると、この問題を容易に解決することができる。遅延評価機能をハイパバイザコールで実現する。

2.3 MMU

カーネルハイパバイザは、ページ・テーブルを持っている。ゲスト OS およびゲスト OS 上のユーザ・プロセスのアドレス空間を、そのページ・テーブルで構成する。すなわち、ゲスト OS のページ・テーブルにある仮想計算機の物理アドレスから、実計算機の物理アドレスを求めて、それを自分自身のページ・テーブルに書き込む。LilyVM では、これまで、TLB のフラッシュを行う時に、ゲスト OS のページ・テーブルをスキャンし、ホスト OS の mmap() システム・コールを用いて、ホスト OS が持つページ・テーブルへの反映を行っていた。しかし、カーネルハイパバイザを用いることで、ページ・フォールトの処理が高速になるため、これ以外の方法を検討している。例えば、TLB のフラッシュを行う時には、カーネルハイパバイザのページ・テーブルを、すべてページ不在とし、アクセスされたときにはじめてゲスト OS のページ・テーブルを参照し反映する方法を検討している。

3. 実　　験

現在、カーネルハイパバイザの一部が動作している。そこで、性能を測定し従来の LilyVM との比較を行った。環境は、CPU が Pentium 4 3.00GHz、ホスト OS は Linux 2.4.27、ゲスト OS は NetBSD 2.0 である。実験では、GNU awk 3.1.4 のコンパイルを行う make コマンドの実行時間を測定した。GNU awk 3.1.4 は、ヘッダファイルとソースファイルが約 140 個あり、行数の合計は約 60,000 行である。結果を**表 1** に示す。このように、従来の LilyVM と比べて約 30% 性能が向上している。ただし、これは、ホスト OS のレベルで VM プロセスに割り当てられたメモリがページ・アウトされないという条件の下で実験した結果である。今後は、ページ・アウトされても動作するようにしたいと考えている。

4. ま と め

本論文では、カーネルレベルで動作するハイパバイザと、ユーザレベルで動作するハイパバイザの 2 つのハイパバイザで仮想計算機を高速化する方法を提案した。その特徴は、フォールトおよびトラップのエミュレーションをカーネルハイパバイザで行うことにより、ユーザレベルのハイパバイザのみで実現するよりも高速である点にある。

今後の課題は、実装と、MMU のエミュレーションにおいて、ホスト OS の VM プロセスに割り当てられたメモリがページ・アウトされた場合に対応することである。

参 考 文 献

1) Eiraku, H. and Shinjo, Y.: Running BSD Kernels as User Processes by Partial Emulation and Rewriting of Machine Instructions, *USENIX BSDCon 2003 Conference (BSDCon'03)* (September 2003).

表 1 アプリケーションベンチマークの結果 (GNU awk 3.1.4 のコンパイル時間)

OS/実行環境	秒
NetBSD/実機	13.0
NetBSD/従来方式/Linux	20.4
NetBSD/本方式/Linux	15.5