仮想マシンモニタによるプロセススケジューリング

田所秀和[†] 光来健一[††] 千葉滋[†]

1. はじめに

近年，サーバの利用率を向上させるために，仮想マシンを用いて複数のサーバを一台のマシンに統合することが増えてきている．このようなシステムでは複数の仮想マシンが一台の物理マシンを共有することになるため，システム全体で優先度の高い処理と低い処理を考えることが必要になる．例えば，バックアップやウィルススキャン等はサーバの提供するサービスを阻害しないように優先度を低くしておき，システム全体の負荷が低い時に動かすべきである．それらのプロセスを動かす仮想マシンの負荷が低いというだけでは不十分で，他に負荷の高い仮想マシンがある場合にはそのサービスを阻害してしまう．

しかし，このようなシステム全体でプロセスを意識したスケジューリングを行うのは難しい．仮想マシン上で動くゲスト OS 内のスケジューリングだけでは，異なる仮想マシン上で動くプロセスに優先度をつけることができない．仮想マシン自体のスケジューリングを併用すれば仮想マシン間でプロセスの優先度をある程度制御することができるが，ゲスト OS のスケジューリングが少し変わるだけでうまく制御できなくなる．例えば，仮想マシン内のプロセスの 1 つが停止するだけで，その仮想マシン内の他のプロセスに割り当てられる CPU 時間が増え，システム全体の中での優先度が上がってしまう．

2. 提案と実装

このような問題を解決するために，本論文では仮想マシンモニタが仮想マシン間にまたがるプロセススケジューリングを行うシステムを提案する．このシステムでは，仮想マシンモニタから仮想マシン上のゲスト OS のランキューを操作することで，ゲスト OS のプロセススケジューリングを調整する．さらに，仮想マシンのスケジューリングおよびゲスト OS の既存のスケジューリングと連携することにより，システム全体でプロセスに優先度をつけることができる．

2.1 システムの概要

本システムでは，仮想マシンモニタとして Xen 仮想マシンモニタ[1] を用いる．Xen 上の仮想マシンはドメインと呼ばれ，ドメインには特権仮想マシンであるドメイン 0 とそれ以外のドメイン U がある．またゲスト OS としては Linux と Windows Vista SP1 を用い，アーキテクチャは `x86_64` を対象としている．

本システムでは，**図 1** のようにスケジューラを Xen 仮想マシンモニタのドメイン 0 上のプロセスとして実装する．このプロセスは仮想マシンのスケジューリングを設定し，さらに一定時間ごとに全てのドメイン U を一時停止し，ドメイン U のゲスト OS のメモリを調べる．ゲスト OS のランキューが一貫性を保って操作可能ならスケジューリングポリシーに応じてランキューを操作してから，ドメイン U を再び動かす．

2.2 ドメイン U のカーネルメモリへのアクセス

Xen は各ドメインにメモリを割り当てるために，マシンメモリと疑似物理メモリを管理する．マシンメモリは物理メモリであり，マシンフレームと呼ばれるページサイズのかたまりに分割される．マシンフレームにはマシンフレーム番号と呼ばれる 0 から始まる番号がつけられる．一方，疑似物理メモリは，ドメインが仮想的に物理メモリとして扱うメモリであり，疑似物理フレームと呼ばれるページサイズのかたまりに分

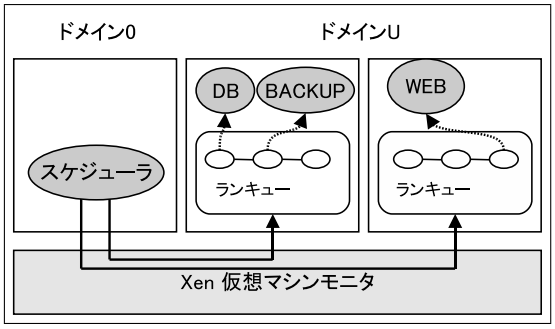

図 1 システムの構成
Fig. 1 An overview of our system.

† 東京工業大学 情報理工学研究科 数理・計算科学専攻
†† 九州工業大学 情報工学研究院 情報創成工学研究系

割される．疑似物理フレームには疑似物理フレーム番号と呼ばれる 0 から始まる番号がつけられる．Xen はマシンフレームと疑似物理フレームの対応を管理することで，ドメインのメモリを管理する．

ドメイン 0 のプロセスからドメイン U のカーネルメモリを読み書きするために，Xen 仮想マシンモニタの機能を用いる．Xen では**図 2** のように，任意のマシンフレームをドメイン 0 のプロセスのアドレス空間にマップすることが可能である．まず，Xen の API を使って，ドメイン U のカーネル空間の仮想アドレスからそのアドレスに対応する疑似物理フレーム番号を求める．次に，その疑似物理フレーム番号に対応するマシンフレーム番号を求め，そのフレームをマップすることで，任意のドメイン U の任意の仮想アドレスを読み書きすることが可能である．

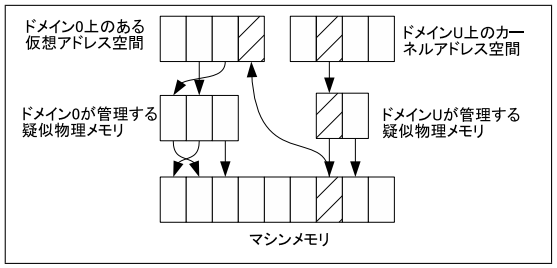

図 2 ドメイン 0 のプロセスにドメイン U のカーネルメモリをマップ
Fig. 2 Mapping DomainU's kernel memory into the address space of a Domain0's process.

2.3 Windows カーネルの操作

カーネルメモリを操作するために、型情報が必要である。本システムでは、Windows 用カーネルデバッガを用い、必要な型情報を事前に取得しておく[2)]。

型情報を使い、プロセスと思われるメモリ上の位置を特定する。Windows 内部では、プロセスなどはオブジェクトとして表現されている。オブジェクトの先頭には型を表すヘッダがついており、プロセス型を表すヘッダのビット列を探索することで、プロセスを表しているオブジェクトの位置を特定する。

プロセスから、そのプロセスが所有するスレッドをたどることができる。このスレッド構造体の優先度フィールドを書換えることで、スレッドの優先度を変更する。

さらに、ランキューにつながれたスレッドを操作することで、スレッドの実行を止めたり再開させる。止めたいスレッドは、ランキューからはずすことで、止めることができる。再開したいばあいは、ランキューにつなぎなおすことで、動かすことができる。

2.4 実　　験

本システムを用いて，他の仮想マシンのすべてのプロセスが停止した時だけ特定のプロセスを動かすというスケジューリングポリシーを実装し，実験を行った．実験の結果，ポリシー通りにプロセスがスケジューリングされることを確認した．

3. まとめと今後の課題

本論文では，仮想マシンモニタによるプロセススケジューリングを提案した．Xen 仮想マシンモニタを用い，ドメイン 0 からドメイン U 上のゲスト OS のメモリを操作する．メモリ操作によって，ゲスト OS のスケジューラのランキューを操作してゲスト OS のプロセススケジューリングを変更する．また，この手法を用いて優先度を変更するスケジューラを実装し，実際にスケジューリングが行えることを確認した．スケジューラはドメイン 0 のプロセスとして実装したので，仮想マシンモニタやドメイン U のゲスト OS のカーネルは変更していない．

今後の課題としては，プロセスの実行時間を監視して，実行時間に応じてプロセスの実行を調整するスケジューリングの実現が挙げられる．プロセスの実行時間の監視や実行の制御は，すでに実装した機構の組み合わせで実装可能である．実際にこのようなスケジューラを実装し，実験を行う必要があると考えている．

また，I/O バウンドなプロセスの調整も行えるようにする必要がある．本システムではプロセスを停止させる方法として，ランキューからプロセスに対応するタスク構造体を取り除くという方法を用いたため，プロセスを止めるにはタスク構造体がランキュー内にある必要がある．I/O バウンドなプロセスはランキューに存在することが少なく，ランキューを操作する方法での制御が難しい．このようなプロセスは，I/O リクエストをドメイン 0 で調整することにより制御することが考えられる．

参 考 文 献

1) Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I. and Warfield, A.: Xen and the Art of Virtualization, *Proc. Symp. Operating Systems Principles*, pp. 164–177 (2003).
2) Mark E. Russinovich, David A. Solomon: Microsoft®Windows®Internals, Fourth Edition: Microsoft Windows Server™2003, Windows XP, and Windows 2000.