B-006

KVM における優先度制御法の有効性
Effectiveness of Priority Control Method in KVM

森山 英明† 山内 利宏‡ 谷口 秀夫‡
Hideaki Moriyama Toshihiro Yamauchi Hideo Taniguchi

1. はじめに

1 台の計算機上で複数のオペレーティングシステム（以降，OS と略す）を同時走行させることで，計算機の資源を効率的に利用する方式が研究され，実用化されている．このような方式の 1 つとして，KVM を用いた仮想計算機方式がある．この方式では，ホストとなる OS に KVM を導入することでハイパーバイザを構成し，複数の OS を仮想計算機として同時に走行させることができる．仮想計算機環境を利用することで，計算機資源の効率的な利用が可能となり，さらに，クラウド環境と組み合わせて負荷分散を行うことで，効率的なデータ処理を実現するシステム[1]も研究されている．ハイパーバイザは，各仮想計算機をプロセスとして登録し，OS 上で動作する他のプロセスと同様にしてスケジューリングし管理する．このため，各仮想計算機に相当するプロセスの優先度を変更することで，各仮想計算機が利用する計算機資源の割り当てを制御することができる．

本稿では，KVM を用いた仮想化環境において，各 OS の優先度を変更したときの影響を示し，有効性について述べる．

2. KVM

KVM（for Kernel-based Virtual Machine）は，Linux カーネル内に VMM（Virtual Machine Monitor）を実現し，VM（Virtual Machine）1 個を 1 つのプロセスとして管理する方式である．なお，KVM における仮想計算機環境は，QEMU と呼ばれるエミュレータにより提供される．

VM を 1 台起動するごとに，qemu-kvm という名前のプロセスが生成される．qemu-kvm プロセスは，生成後，仮想プロセッサの作成，メモリ資源の割り当て，および初期化といった処理を行う．これらの処理によって仮想化環境の準備を行い，終了後に OS の初期化処理を開始する．なお，VM の実行が終了すると，qemu-kvm プロセスは終了処理を行った後に消滅する．

3. 評価

3.1 目的

KVM による仮想計算機の処理負荷を測定する．また，各仮想計算機に相当する qemu-kvm プロセスについて，プロセスの優先度を変更しない場合と変更した場合とで測定を行い，優先度の変更による処理の影響を示す．

† 有明工業高等専門学校
Ariake National College of Technology

‡ 岡山大学大学院自然科学研究科
Graduate School of Natural Science and Technology,
Okayama University

表 1 評価計算機

CPU	Intel(R) Core(TM) i3-3220 (3.30GHz)
メモリ	4GB
OS	Fedora14
QEMU	0.13.0

3.2 評価環境

本評価で使用した計算機の環境を表 1 に示す．KVM を利用するために，CPU は Intel VT-x の機能を持つ Core i3 を用いる．コア数について，評価に用いる Intel Core i3 の物理プロセッサ数は 2 個であるが，ハイパースレッディング機能を有効にしているため，論理プロセッサコア数は 4 個となる．なお，本評価で用いるゲスト OS はすべて同じ構成にしており，OS を CentOS 6.4，コア数を 1 としている．

3.3 評価プログラム

評価プログラムとして，1 から 10,000,000 までの整数値に対して，1 つずつ素数であるか否かを判断し，素数である場合はその値を標準出力するプログラムを作成した．評価プログラムは，ゲスト OS 上の/etc/init.d 上に起動用のスクリプトを用意し，ゲスト OS の初期化終了後にデーモンとして動作するよう設定する．なお，ホスト OS 上でゲスト OS を起動せずに評価プログラムを実行した場合，プログラムの実行開始から終了までの実行時間は 11.9 秒，ユーザモードでの実行時間は 8.1 秒，およびカーネルモードでの実行時間は 3.7 秒となった．

3.4 評価方法

測定では，複数のゲスト OS 上で評価プログラムを実行し，評価プログラムの実行時間を測定する．実行時間は，プログラムの開始から終了までの実行時間（以降，real），ユーザモードでのプログラムの実行時間（以降，user），およびカーネルモードでのプログラムの実行時間（以降，sys）を測定する．また，各ゲスト OS に相当する qemu-kvm プロセスについて，ゲスト OS 起動時から評価プログラム終了時までの CPU 使用率を測定する．具体的には，仮想化環境の準備，OS の初期化，評価プログラムの実行における CPU 使用率を測定する．

本評価では，以下の 3 つの構成について測定を行った．

(構成 1) 単一のゲスト OS 上で評価プログラムを動作させた場合（プロセス優先度の変更なし）

(構成 2) 5 個のゲスト OS 上で評価プログラムを動作させた場合（プロセス優先度の変更なし）

(構成 3) 5 個のゲスト OS の内 1 つのプロセス優先度を他よりも高く設定し，評価プログラムを動作させた場合

表 2 実行時間の測定結果 (s)

構成	time	ゲスト OS1	ゲスト OS2	ゲスト OS3	ゲスト OS4	ゲスト OS5
構成 1	real	49.0				
	user	8.3				
	sys	34.0				
構成 2	real	189.6	176.1	178.2	188.8	182.2
	user	14.8	12.9	13.0	14.1	13.8
	sys	157.5	150.1	154.0	159.1	157.0
構成 3	real	166.4	168.8	182.3	185.4	143.5
	user	13.8	13.2	14.5	14.5	10.7
	sys	141.2	143.1	153.0	157.2	121.4

表 3 構成 2 と構成 3 の実行時間の比較結果 (%)

time	ゲスト OS1	ゲスト OS2	ゲスト OS3	ゲスト OS4	ゲスト OS5
real	−12.20	−4.12	2.30	−1.80	−21.23
user	−6.51	2.80	12.04	2.84	−22.36
sys	−10.36	−4.66	−0.66	−1.18	−22.64

構成 1 は，KVM が動作するホスト OS において，qemu-kvm プロセスが十分に CPU を利用できる場合である．構成 2 と構成 3 は，qemu-kvm プロセスが十分に CPU を利用できない場合である．また，構成 2 ではプロセス優先度を考慮せずにゲスト OS を実行するのに対して，構成 3 ではゲスト OS 5 のプロセス優先度を高く設定して実行する．プロセス優先度の設定について，Linux のスケジューラでは，各プロセスに nice 値が設定されている．nice 値は−20 から+19 までの範囲から値をとり，優先度の高いプロセスほど小さい nice 値が設定される．スケジューラは，各プロセスに設定されている nice 値より，プロセスの静的優先度と動的優先度を算出し，優先度に応じたスケジューリングを行う．なお，ユーザプログラムの場合，指定がなければ初期値として 0 が設定される．すなわち，測定の構成 1 と構成 2 では，各ゲスト OS に対応する qemu-kvm プロセスの nice 値は 0 に設定する．構成 3 では，ゲスト OS 1〜4 に対応する qemu-kvm プロセスの nice 値は 0 とし，ゲスト OS 5 に対応する qemu-kvm プロセスの nice 値は−20 と設定する．

3.5 測定結果

構成 1 から構成 3 について測定した結果を表 2 に示す．構成 1 の測定において，user が 8.3 秒であることから，3.3 節で述べたホスト OS 上で評価プログラムを実行したときのユーザモードの実行時間に近い値を得ることができた．しかし，sys は 34.0 秒であり，およそ 10 倍近く実行が遅くなっている．この理由は，評価プログラムでは，素数の計算後に標準出力で素数の表示処理を行っており，表示処理に必要となる write システムコールの実行のオーバヘッドが大きくなっているためと考えられる．構成 2 では，ゲスト OS を 5 個にしたことにより，各 qemu-kvm プロセスに十分な CPU を割り当てることができない．このため，実行時間を構成 1 と比較すると，user は約 6 秒，sys は約 120 秒長くなる．

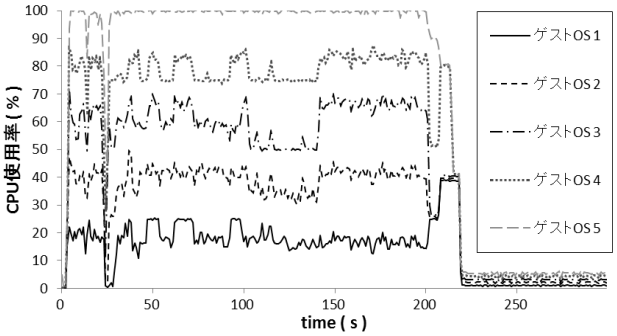

図 1 構成 2 の CPU 使用率の測定結果

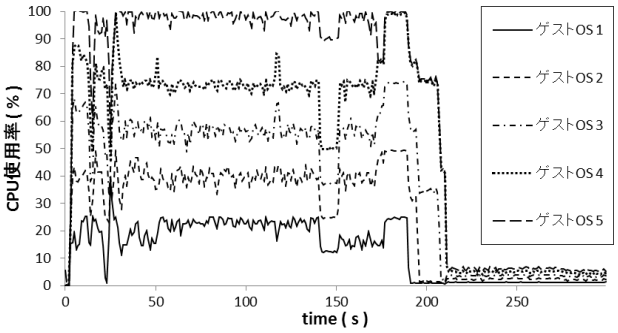

図 2 構成 3 の CPU 使用率の測定結果

構成 2 と構成 3 の差分から実行時間の短縮の割合を出したもの（(構成 2 の実行時間−構成 3 の実行時間) ÷ 構成 2 の実行時間×100 (%)）を表 3 に示す．構成 3 の実行時間が構成 2 よりも増加する場合は増加の割合を正の値で，減少する場合は減少の割合を負の値で示している．表 3 より，ゲスト OS 1〜4 の実行時間短縮の割合は最大でも約 12%である．これに対して，ゲスト OS 5 では real, user，sys において約 21%短縮することができる．

構成 2 と構成 3 について， 各ゲスト OS の CPU 使用率の測定結果を，それぞれ図 1 と図 2 に示す（各ゲスト OS の CPU 使用率を積み上げて示す）．図 1 より，プロセスの優先度を設定しない場合，各ゲスト OS は約 20%の CPU 使用率で動作する．一方，図 2 より，優先度を高く設定したゲスト OS 5 は，約 25%の CPU 使用率で動作する．

以上より，KVM において，ゲスト OS に相当する qemu-kvm プロセスの優先度を高く設定することで，そのプロセスが使用する CPU の割り当て頻度を高くすることができ，ゲスト OS 上の処理時間を短縮することが可能となる．

4. おわりに

KVM を用いた仮想計算機方式において，ゲスト OS のプロセス優先度を変更することによる処理の影響について評価した．測定の結果より，ゲスト OS へ十分な CPU 割り当てができない場合において，プロセスの優先度を高くすることで CPU の割り当ての頻度を高くすることができ，実行時間を短縮することができた．残された課題は，CPU 以外の資源における優先度制御の評価がある．

参考文献

[1] 笠江 優美子，小口 正人：ハイブリッドクラウド環境における性能と実行コストに基づいたデータ処理最適配置ミドルウェアの提案と実装，電子情報通信学会技術研究報告，vol.112, no.346，pp.143-148(2012.12).