

CPU 資源の割当て制御による KVM の動作制御評価

Evaluation of CPU resources allocation control in KVM

森山 英明† 木下 椋司† 菅沼 明† 山内 利宏‡ 谷口 秀夫‡
Hideaki Moriyama Ryoji Kinoshita Akira Suganuma 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. この方式の一つとして, KVM (for Kernel-based Virtual Machine) を用いた仮想計算機方式がある. 仮想計算機方式の一つとして, 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ることで, 効率的なデータ処理を実現するシステムも研究されている. また, 近年では, Cgroups (Control Groups) と呼ばれる技術が提供されている. これは, プロセスのグループごとに資源の割当てを制御する機能であり, 割当て精度などに関する研究[1]など, 様々な研究がされている.

本稿では, VM (Virtual Machine) が利用する CPU 資源に着目し, オペレーティングシステムのスケジューラで用いる優先度 (nice 値) を用いた制御と Cgroups を用いた制御の 2 つから制御の効果を確認し, 制御の有効性に関する評価を行った結果を報告する.

2. プロセス優先度と Cgroups による制御

2.1 プロセス優先度による制御

KVM と呼ばれる仮想化技術は, Linux カーネル内にハイパーバイザを実現し, VM1 個を 1 つのプロセスとして管理する方式である. 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, 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.

また, Linux におけるスケジューラは, 各プロセスに設定されている優先度に基づくスケジューリングを行う. 特に, 静的優先度の計算で用いられる 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.

以上より, プロセス優先度を用いた制御では, 各仮想計算機に相当する qemu-kvm プロセスに対して, 割り当てる CPU 使用率に応じた nice 値を設定することで, 各仮想計算機への CPU 割り当てを制御する.

2.2 Cgroups による制御

Cgroups は, Linux 2.6.24 以降にカーネルの機能として提供されており, プロセスのグループに対する資源管理を可能とする. 階層構造で管理されるプロセスの各グループに対して, 資源を割り当てる比率を設定することで CPU, I/O, メモリ, ネットワークなどの資源割当て管理を可能とする. 割り当ての比率は, 各プロセスグループの cpu.shares で管理され, 初期値は 1024 である.

3. 評価

3.1 目的

評価では, KVM による仮想計算機の CPU 処理の負荷を測定する. また, プロセス優先度と Cgroups による制御に対して測定を行い, 制御を加えない場合との比較と各制御の比較を行う.

3.2 評価方法

本評価では, KVM を利用するために, CPU は Intel VT-x の機能を持つ Intel(R) Core(TM) i3-3220 (3.30GHz) を用いた. 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. また, ホスト OS は Fedora14 (Linux Kernel 2.6.35), ゲスト OS はすべて同じ構成にしており, OS を CentOS 6.4 (Linux Kernel 2.6.32), コア数を 1 としている.

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.

評価プログラムとして, 1~50,000,000 まで順に素数か否かを判定する CPU バウンドのプログラムを作成した. 測定では, 評価プログラムを各仮想計算機上に用意し, 各仮想計算機上で at コマンドを用いて同時刻に実行を開始するよう設定する. なお, ホスト OS 上で仮想計算機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 76.9 秒, ユーザモードでの実行時間は 76.6 秒, およびカーネルモードでの

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

‡ 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 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	78.3				
	User	78.3				
	Sys	0.0				
構成 2	Real	132.5	106.3	134.5	123.0	102.4
	User	132.3	106.2	134.3	122.9	102.4
	Sys	0.0	0.0	0.0	0.0	0.0
構成 3	Real	124.2	139.7	103.5	125.5	104.5
	User	124.1	139.4	103.4	125.4	104.4
	Sys	0.0	0.0	0.0	0.0	0.0
構成 4	Real	102.3	138.1	135.8	104.4	110.6
	User	102.3	138.0	135.8	104.3	110.5
	Sys	0.0	0.0	0.0	0.0	0.1

実行時間は0.0秒となった。

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

(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

(構成 3) 5 個のゲスト OS の内 1 つ (ゲスト OS1) のプロセス優先度を他よりも高く設定 (nice = -20 に設定) し、評価プログラムを動作させた場合

(構成 4) 5 個のゲスト OS の内 1 つ (ゲスト OS1) に対して cpu.shares の値を高く設定 (2048 に設定) し Cgroups で制御し、評価プログラムを動作させた場合

3.3 測定結果

実行時間に関する測定結果を表 1 に示す。まず、構成 2 はゲスト OS を 5 個動作しており、論理プロセッサコア数はゲスト OS 数に対して不足しているため、ゲスト OS を 1 個のみ動作させて測定したときと比較して、実行時間は長くなっている。次に、構成 2 と構成 3 の実行時間を比較すると、プロセス優先度を高く設定したゲスト OS1 の実行時間が約 10 秒短縮されている。一方、構成 3 ではゲスト OS1 のプロセス優先度を他のゲスト OS よりも高くしているにも関わらず、ゲスト OS3 や 5 よりもゲスト OS1 の実行時間が長い。この理由として、構成 2 の CPU 使用率を表した図 1 と構成 3 の CPU 使用率を表した図 2 を比較すると、ゲスト OS2 と 4 の CPU 使用率が特に低く、ゲスト OS3 と 5 が長い時間 CPU を利用できたことから発生した問題ではないかと考える。最後に、構成 4 の実行時間の測定結果に着目すると、ゲスト OS1 は他のゲスト OS よりも実行時間が短い。CPU 使用率を表した図 3 に着目すると、ゲスト OS1 の CPU 使用率は、常に高く割り当てられている。

以上より、今回の測定では、プロセス優先度による制御よりも Cgroups を用いた制御の方が制御の効果が高い。

4. おわりに

本稿では、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制御と

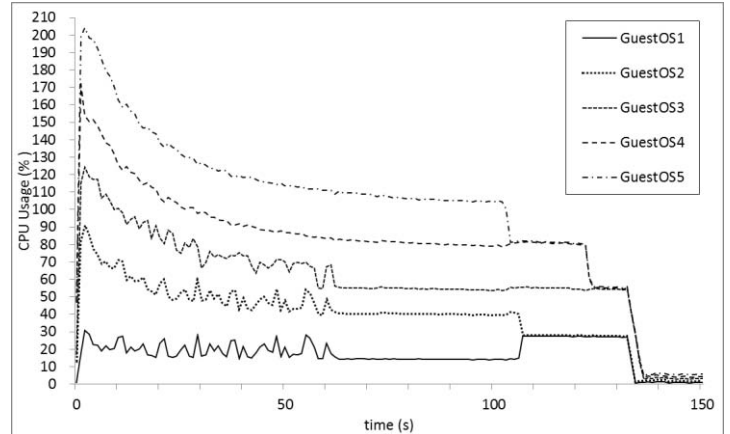


図1 ゲスト OS の CPU 使用率 (構成 2, 制御なし)

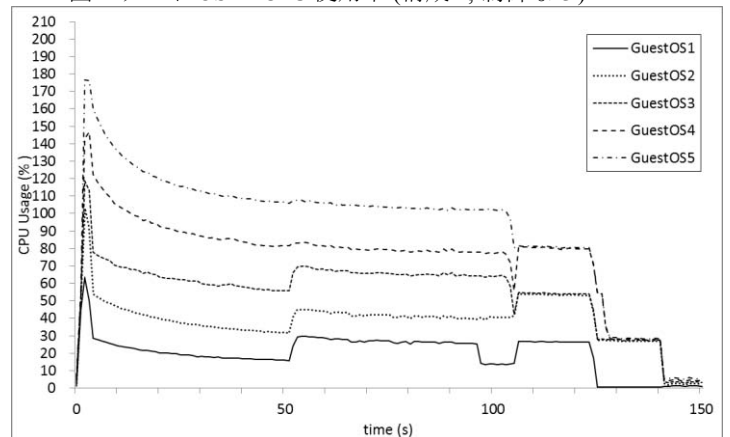


図2 ゲスト OS の CPU 使用率 (構成 3, ゲスト OS1 を高優先)

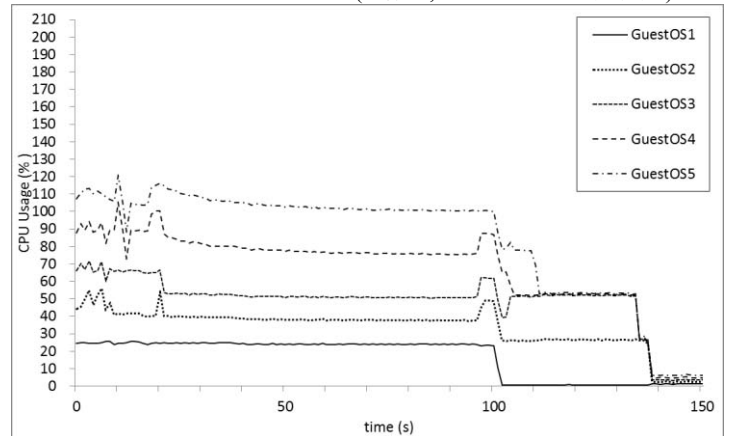


図3 ゲスト OS の CPU 使用率 (構成 4, Cgroups で制御, ゲスト OS1 の設定高)

Cgroups を用いた制御について、評価を行った。測定の結果より、どちらの制御法も、ゲスト OS へ十分な CPU 割り当てができない場合において、CPU の割り当ての頻度を高くすることができ、特に Cgroups を用いた制御の効果は高かった。残された課題は、より多くの仮想計算機を用いた詳細な評価を行うことである。

参考文献

- [1] 富樫壮太, 片山吉章, 出口昌弘, 毛利公一: Linux の資源管理機構 Control Groups の性能調査, 情報処理学会全国大会講演論文集 2012, vol.1, pp.125-127, 2012.