6A-4

仮想化ソフト上で動作するゲストOSのリアルタイム性評価

井邊　研吾†　　攝津　敦†　　落合　真一†

三菱電機株式会社　　情報技術総合研究所†

1. はじめに

仮想化技術は既存システムを従来のOSごと移行する事を可能にする技術である。我々はリアルタイム性が要求される産業用システムに対してこの技術の適用可能性を検討している。仮想マシン上の環境はリアルタイム性の低下が考えられるため評価と改善が必要となる。

リアルタイム性阻害要因を探るべく、仮想マシンのOSとそのI/Oに負荷をかけて周期起床するプログラムの応答遅延を評価した。本稿は仮想マシンを用いない環境で応答遅延が5ms程度で収まるのに対し、仮想マシンを用いた環境においてどの程度遅延が発生するかを測定している。この測定結果と改善案について述べる。

2. リアルタイム性評価

2.1. 評価構成

本稿における評価は仮想環境としてKVM(Kernel-based Virtual Machine)[1]を利用している。KVMはqemu-kvm[2]を用いることでH/Wエミュレーションを行っている。本稿では仮想環境を提供するOSをホストOS、仮想環境上で動作するOSをゲストOSと呼ぶ。評価で用いた構成を表 1、表 2に示す。

表 1 H/W構成

CPU	Core i5-2400 3.1GHz
メモリ	8GB(内1GB ゲストOSで使用)

表 2 S/W構成

ホストOS	OS	64bit CentOS 6.3
	カーネル	2.6.32 PREEMPT
	qemu-kvm	0.12.1
ゲストOS（評価対象）	OS	32bit CentOS 6.3
	カーネル	2.6.32 PREEMPT

本稿が想定している既存システムは32bitのため、ゲストOSを32bitシングルコアとし、CPUコア1つを割り付けて評価を行った。(図 1)

2.2. 評価方法

評価プログラムは周期起床プログラムと負荷プログラムから成る。周期起床プログラムはシステムにおけるリアルタイムアプリケーションを模擬するものであり、システムタイマを用いて周期的に起床し、想定した起床時刻と実際の起床時刻のジッタを測定する。(図 2)

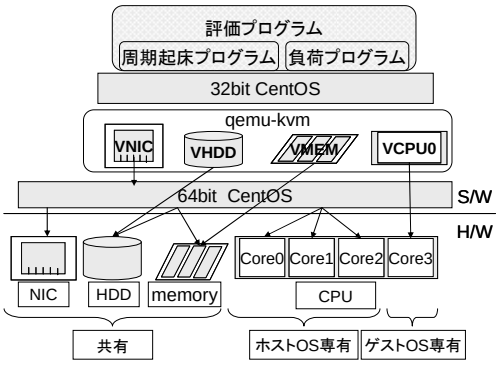

図 1 評価構成

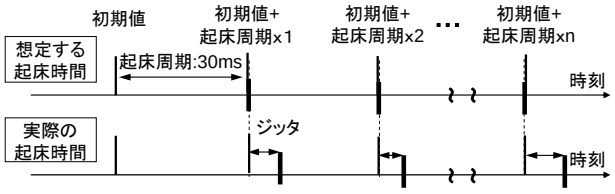

図 2 周期起床プログラムの動作

負荷プログラムはカーネル負荷とI/O負荷の2つから構成される。今回の評価ではゲストOS上のみ負荷プログラムを実行した。

(1)カーネル負荷：

カーネル負荷として共有メモリとメモリへのアクセス、スケジューラへの負荷・タイマ割込み・セマフォの獲得と解放を行う。

(2)I/O負荷：

ディスクへの負荷として読書き・Syncによる強制書込み・ファイルの作成と削除を行う。LANへの負荷としてホストOSとゲストOS間でUDP通信を用いてパケットの送受信を行う。

3. 評価結果

はじめに、カーネル負荷のみをかけて評価を行った。評価の結果、プロセスの起床24万回中、最大の遅延が1.16msに収まることを確認した。

さらにカーネル負荷に加えI/O負荷も与え評価を行った。その結果100ms以上の遅延が発生することを確認した(図 3)。この結果から仮想環境の性能面で課題とされるI/O処理がリアルタイム性においても課題となることを確認した。

表 3 測定結果

カーネル負荷のみ	カーネル+I/O負荷
1.16ms	100.55ms

Evaluation of Realtime Performance of an Operating System in Virtualized Environment
Kengo Ibe, Atsushi Settsu, Shinichi Ochiai
†Information Technology R&D Center, Mitsubishi Electric Corporation

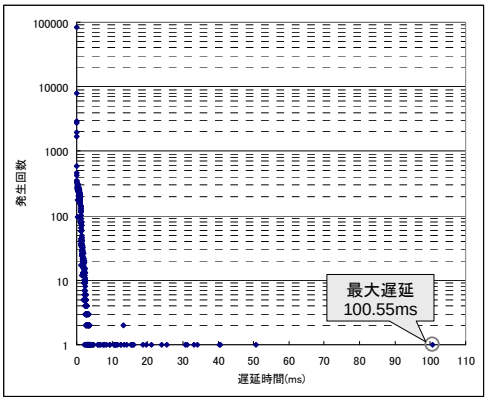

図 3 I/O 負荷による遅延

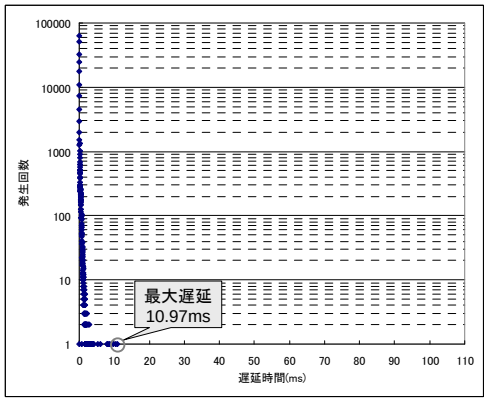

図 4 解決策適用後の遅延

4. 考察

ゲスト OS にカーネル負荷のみをかけた場合とゲスト OS に I/O 負荷もかけた場合の差異は、ホスト側へ制御が遷移するか否かという点と考える。カーネル負荷のみの場合、ゲスト OS 内で処理が完結するが、I/O 負荷を加えると、ホスト側にある qemu-kvm による H/W エミュレーション処理が発生する。H/W エミュレーション時はゲスト OS が停止状態となり、これがリアルタイム性の低下を引き起こす要因の 1 つであると考える。このリアルタイム性低下の要因について考察する。

4.1. エミュレーション処理

エミュレーション処理はゲスト OS がレジスタアクセスするたびにホスト側に制御が移行し、その間ゲスト OS が停止状態となる。

4.2. I/O アクセスの競合

I/O をホストと共有しているため、ホスト OS との I/O の競合が発生し、I/O の処理待ちが発生する。その結果、qemu-kvm の I/O 処理が遅延し、ゲスト側の I/O 処理の時間が増加すると考えられる。

5. 解決策

4.1、4.2節の考察に対し解決策を考案する。

課題	解決策
エミュレーション処理	virtio・vhost-net の適用による オーバヘッドの削減
I/O アクセスの競合	qemu-kvm の RT 優先度実行

5.1. virtio・vhost-net の適用

エミュレーションによるオーバヘッドを削減するため、準仮想化でI/Oにアクセスすることが可能なvirtio[3]を適用する。

さらに、virtioを適用したネットワークに対し、qemu-kvmのプロトコル制御の処理をホストOS内で行い、制御の遷移に要するオーバヘッド削減することが可能なvhost-net[4]を適用する。この 2 点を適用し、I/Oアクセスに要するオーバヘッドを削減することで、リアルタイム性が向上すると考えられる。

5.2. qemu-kvm を最高優先度プロセスに設定

ホストとの I/O の競合を避けるため、ゲスト側とホスト側で I/O を個別に用意し、I/O を分離する事が考えられる。しかし、今回のように分離できない場合には、qemu-kvm プロセスを最高優先度に設定することで、ゲスト側の I/O 処理を優先的に処理し、競合時の待ち時間を減らす方法が有効であると考える。

6. 解決策適用後の評価結果

5.1、5.2節の解決策を適用し評価を行った結果10.97msまで遅延を抑えることができた(図 4)。

7. 結論

ゲスト OS とその I/O に負荷をかけて周期プログラムの遅延を測定したところ、解決策適用前では最大 100.55ms の遅延が発生した。

これに対し、解決策を適用し、前回と同様の評価を行った結果、10.97ms まで遅延を抑えることができた。このことから、解決策である優先度制御・virtio・vhost-net はリアルタイム性向上に有効であると考える。

8. 今後の課題

ある程度のリアルタイム性の向上を実現したが、仮想マシンを用いない環境では 5ms 程度であり、10ms 程度の遅延では依然として改善が必要であると考えている。また、ホスト側の負荷も考慮にいれたリアルタイム性阻害要因の分析が必要であると考える。

今後、qemu-kvm やカーネルを改良することも視野に入れリアルタイム性向上の改善方法について検討していく予定である。

参考文献

[1] Kernel Based Virtual Machine
http://www.linux-kvm.org/page/Main_Page
[2] QEMU - KVM
http://wiki.qemu.org/KVM
[3] KVM - Virtio
http://www.linux-kvm.org/page/Virtio
[4] KVM - VhostNet
http://www.linux-kvm.org/page/UsingVhost