Vol.2013-ARC-205 No.8
Vol.2013-OS-125 No.8
2013/4/26

Mintオペレーティングシステム上のKVMの評価

仲尾 和祥[1]　乃村 能成[1]　谷口 秀夫[1]

概要：我々は，1 台のマルチコア計算機で複数 Linux を独立に走行させる方式の OS として，Mint オペレーティングシステムを開発している．本稿では，Mint で走行する各 Linux に仮想計算機方式の KVM を導入し，1 台の計算機上で複数のハイパーバイザを走行させる方式について述べる．これにより，ハイパーバイザ間の処理負荷の影響を軽減しつつ，各ハイパーバイザ上の OS 間では分割された資源を共有して利用できる．また，ハイパーバイザの障害時の影響を限定でき，Mint 上で Linux 以外のカーネルを走行可能にする．基本評価では，Mint 上の KVM と通常の Linux の KVM の性能が同等であることを示した．また，KVM を用いて複数の OS を同時走行させる場合，提案方式を利用して KVM 上で動作させる OS を分散させることで処理負荷の影響が軽減できることを示した．具体的には，I/O 処理の処理負荷の影響を限定でき，レジスタ操作処理とデータ複写処理の性能は通常の Linux で動作する KVM と同等であることを示した．

1. はじめに

計算機資源を効率的に利用する方式として，1 台の計算機で複数のオペレーティングシステム (以後，OS と呼ぶ) を同時走行させる方式が活発に研究されている．この方式の 1 つとして，Xen[1]，KVM[2] および VMware Workstation[3] といった仮想計算機方式 (以後，VM 方式と呼ぶ) がある．

VM 方式では，ハイパーバイザ上で走行させる計算機資源を複数の OS が共有して利用できる利点がある．しかし，OS 間には処理負荷の相互影響があるほか，仮想化によるオーバヘッドが存在する．また，計算機を論理分割し，複数の OS を走行させる Virtage[4] がある．Virtage は OS が CPU コアとデバイスを物理的な粒度で分割する占有モード，または動的に処理実行時間を割り当てることで OS 間で計算機資源を共有する共有モードにより，複数の OS を 1 台の計算機上で動作させる．そして，Virtage 上で動作させた OS 上で KVM を使用することで，複数の KVM を同一計算機上で動作させる方式がある．しかし，Virtage では，入出力デバイスの利用が CPU の仮想化支援機構に依存する形で設計されている．具体的には，Intel の VT-x によってセンシティブ命令について解析を行い，VT-d を使用してデバイスドライバがゲスト OS が使用しているメモリに直接データを書き込めるようになっている．

一方で，我々は，1 台のマルチコア計算機で複数 Linux を独立に走行させる方式の OS として，Mint オペレーティングシステム (以後，Mint と呼ぶ) を開発している [5]．Mint では，CPU コア，実メモリ，および入出力デバイスを分割し，これらを各 OS で占有することで，複数 Linux の同時走行を実現している．しかし，CPU コアや入出力デバイスの分割の最小単位に物理的な制約がある．具体的には，Mint で走行する各 Linux(以後，OS node と呼ぶ) は，CPU をコア単位で分割して占有するため，コア数を越えた OS node を起動できない．また，入出力デバイスをデバイス単位で分割して占有するため，筐体に接続可能なデバイスの上限で OS の構成が制限される．

そこで，Mint で走行する各 Linux に KVM を導入し，1 台の計算機上で複数のハイパーバイザを走行させる．これにより，ハイパーバイザ間の処理負荷の影響を抑制しつつ，各ハイパーバイザ上の OS 間では分割された資源を共有し，柔軟に利用できる．また，ハイパーバイザの障害時の影響を限定するほか，動作可能な OS を多彩化できる．

本稿では，Mint において複数ハイパーバイザを走行させる方式を提案し，この方式の評価について述べる．

2. Mint オペレーティングシステム

2.1 概要

Mint は仮想化によらず，1 台の計算機上で複数の Linux を独立に走行させる方式の OS である．Mint の設計方針として，以下の 2 つがある．

(1) 走行している全ての Linux が相互に処理負荷の影響を与えない．

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

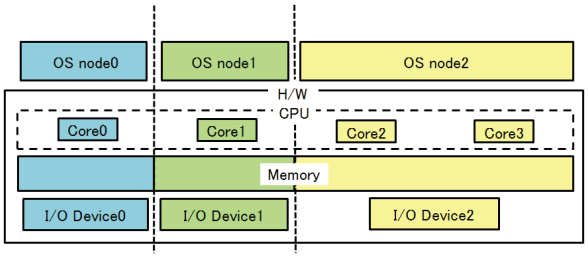

図 1 Mint の構成例

(2) 全ての Linux が入出力性能を十分に利用できる.

Mint の構成例を**図 1** に示す. Mint では, 最初に 1 つの Linux を起動し, この Linux が別の Linux を起動する. なお, Mint では, 最初に起動する Linux を OS node0 とし, 後から起動する N 番目の Linux を OS nodeN とする. また, Mint では, 計算機資源を論理分割して複数の OS node が独立に走行している. 具体的には, CPU はマルチコアプロセッサをコア単位で分割し, 各 OS node は 1 つ以上のコアを占有する. メモリは実メモリ領域を空間分割し, 各 OS node に分配する. 入出力デバイスはデバイス単位で分割し, 各 OS node は指定された入出力デバイスのみを占有する.

図 1 では, OS node0 は CPU の Core0 とメモリの一部, および I/O Device0 を占有して動作する. その後, OS node1 は CPU の Core1 とメモリの一部, および I/O Device1 を占有して動作し, OS node2 は CPU の Core2 と Core3, メモリの一部, および I/O Device2 を占有して動作する.

2.2 機能

Mint は, 利用可能なデバイスを仮想化によらず分割して実現しているため, 既存 (オリジナル) の Linux とほぼ同等の機能を有している. そのため, オリジナルの Linux が持つ各種機能を利用できる. たとえば, CPU ホットプラグ, メモリホットプラグ, および Loadable Kernel Module (LKM) によるデバイスホットプラグがある. また, 各 OS node 間には主従関係がないため, 全 OS node は, 単体での再起動や, 新たな OS node の起動が可能である. 以下に Mint が持つ代表的な機能について概要を説明する.

(1) 計算機資源の再分配機能

Mint では, 動作する OS node 間において, 起動後に CPU, メモリ, および入出力デバイスを動的に再分配できる. それぞれの再分配法について以下に示す.

(a) CPU

オリジナルの Linux には, CPU ホットプラグ機能があり, これにより, 起動後の Linux において, 動的に CPU コアを着脱できる. Mint では, この CPU ホットプラグ機能を応用し, 各 OS node は CPU コアの解放や未使用コアの占有機能を持つ. これによりシステム全体で CPU コアの再分配を可能としてる.

(b) メモリ

Mint では, オリジナルの Linux が持つメモリホットプラグ機能を応用し, 各 OS node が実メモリの解放と占有を行うことで, メモリの再分配を可能としている.

(c) 入出力デバイス

Mint では, オリジナルの Linux と同様, LKM によりデバイスドライバを動的に着脱できる. すなわち, OS node1 が利用中のデバイス devX について, LKM をアンロードし devX を解放した後, OS node2 が devX 用 LKM をロードすることで, devX を OS 間で移譲できる.

(2) 異なる Linux の共存制御機能

Mint は, 複数の x86 Linux Kernel を 1 つの CPU の異なるコア上で同時に独立走行させる技術であるが, 各 Linux の構成は同一でなくともよい. 具体的には, x86 32bit, 64bit カーネルの同時走行や Linux の亜種である Android の同時走行が可能である. これにより, 例えば, 32bit カーネルでしか対応しないデバイスのために 1 コアだけを 32bit カーネルで走行させられる [6].

(3) 独立性の高いカーネル起動機能

Mint では, Linux の標準機能である Kexec[7] を拡張することで, 他 OS node の起動と各 OS node の単独再起動を可能としている. 具体的には, 通常の Linux は BIOS, ブートローダ, セットアップルーチンを経てカーネル本体の初期化を行う. これに対して, Kexec はこれらの処理を自身で代替し, カーネル本体の初期化を行う. 全 OS node は, Kexcec の機能を利用することで, 他 OS に依存することなく, 単体での再起動や新たな OS node の起動が可能である.

2.3 Mint の制約

現在, Mint の制約として以下の 2 つがある.

(1) 計算機の資源分割の粒度が大きい.

Mint における CPU 分割の最小単位はコア単位であり, 入出力デバイス分割の最小単位はデバイス単位である. このため, CPU コア数を超える数の OS node を走行できない. また, 走行する OS node の数だけ磁気ディスクを用意するなど, 各 OS node 単位で使用する入出力デバイスを用意する必要がある.

(2) 走行させる OS カーネルの改変が必要である.

Mint では, 複数 Linux の同時走行を可能にするため, カーネルに変更を加える必要がある. このため, カーネルに変更を加えることができない OS を走行できない.

3. Mint オペレーティングシステムを利用した複数ハイパーバイザの動作

3.1 構成

Mint を利用して複数ハイパーバイザを動作させる方式を提案する．この構成例を**図 2** に示し，説明する．

この構成例では，OS node0，OS node1 および OS node2 をそれぞれ動作させ，OS node1 と OS node2 では，それぞれハイパーバイザとして KVM を用いて VM を動作させている．この際，OS node0 と OS node1 および OS node2 はそれぞれ独立に動作している．このため，OS node1 上の VM と OS node2 上の VM の間において，相互に影響を受けずに動作し，いずれかの OS node が異常終了した際であっても，他の OS node はその影響を受けずに動作できる．また，OS node1 が占有している I/O Device1 は，OS node1 上で動作している VM 間で共有して利用でき，OS node2 が占有している I/O Device2 は，OS node2 上で動作している VM 間で共有して利用できる．

3.2 期待される効果

Mint を利用して複数のハイパーバイザを動作させることで実現される効果について以下に 4 つ示す．

(1) 堅牢性の向上
Mint は，走行している全ての OS node は独立であるという利点がある．そこで，Mint において複数の OS node を起動し，各 OSnode 上でハイパーバイザを起動し，VM を分散して動作させる．例えば，図 2 において，OS node1 に不具合が生じて異常終了したと仮定する．この場合，OS node1 上の VM は全て停止するが，OS node0 および OS node2 は問題なく動作できる可能性が高い．また，各 OS node が独立して起動と再起動を行えるため，ホスト OS を一度停止させる必要がある場合であっても，計算機全体を停止させる必要がない．

(2) VM の処理負荷による影響の限定
Mint には，走行している OS node 間の処理負荷の影響は僅かであるという利点がある．そこで，複数 OS node を起動し，動作させる VM を OS node ごとに分散させることで，VM の処理負荷による影響を OS node ごとに限定できる．例えば，図 2 において，OS node0，OS node1，および OS node2 は独立に動作している．また，OS node1 上の VM 間では処理負荷の影響が発生する．同様に OS node2 上の VM 間では処理負荷の影響が発生する．しかし，OS node1 上の VM と OS node2 上の VM の処理負荷の影響は抑制できる．

(3) 資源の共有/非共有範囲の限定
Mint では，2.3 節の (1) で述べたように，計算機の資源分割の粒度が大きくなる制約がある．また，VM 方式では，ハイパーバイザが計算機の資源の管理を行うことで，資源を仮想化し，各 VM に動的に割り当てられる．そこで，Mint を利用して各 Linux をハイパーバイザとして動作させることで，ハイパーバイザごとに分割した資源をハイパーバイザ上の各 VM で共有して使用する．これにより，計算機資源をより細かい粒度で分配できる．

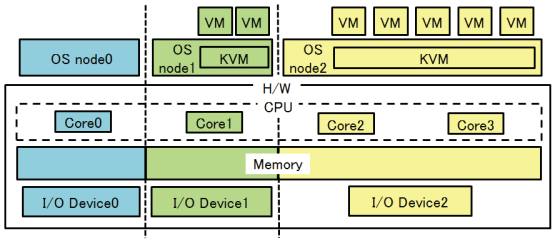

図 2 Mint において複数 VM の動作させた構成例

(4) 動作可能 OS の多彩化
Mint において，2.3 節の (2) で述べたように，走行させる OS のカーネルを改変しなければならない制約がある．そこで，完全仮想化を行うハイパーバイザを利用することで，Mint において改変不要で OS を動作可能にする．例えば，図 2 において，OS node0，OS node1，および OS node2 はカーネルを変更して動作している．しかし，VM は OS のカーネルを変更せずに動作できる．

4. 評価

4.1 目的

本評価の目的を以下に 2 つ示す．

(1) Mint の基本性能を明らかにする．このため，Mint を用いて複数の Linux を動作させた際の性能と KVM を用いて複数の Linux を動作させた際の性能の比較を行う．

(2) Mint で KVM を使用した場合の性能を明らかにする．このため，未改変の Linux(以後，Vanilla Linux と呼ぶ) と Mint を KVM を使用している場合と使用していない場合の性能の比較を行う．

(3) Mint で複数の KVM を使用し，VM を動作させた場合の処理性能について明らかにする．このため，Mint で KVM を使用した場合と Vanilla Linux で KVM を使用した場合の性能の比較を行う．

4.2 評価構成

評価を行う構成を**図 3** に示し，それぞれの構成について以下で説明する．

(構成 1) Vanilla Linux *(Linux)*

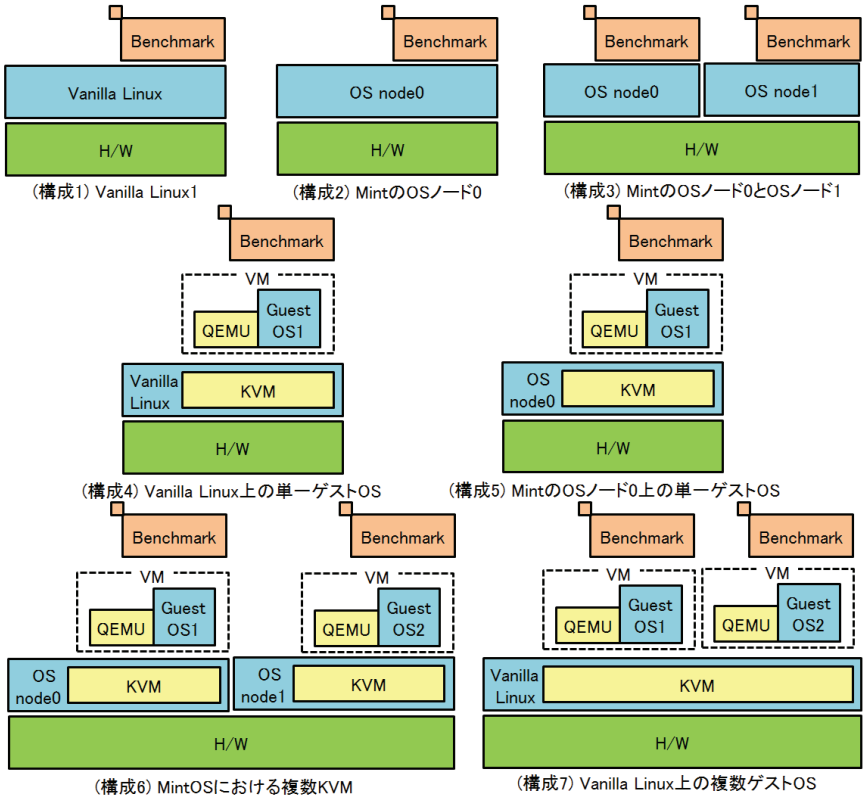

図 3 評価を行う構成

Vanilla Linux のみを動作させる構成である．この構成は，他の構成と比較する際の指標とするために測定する．Vanilla Linux で評価プログラムを動作させる．

(構成 2) Mint の OS node0 *(Mint0)*

Mint において OS node0 のみを動作させる構成である．この構成は，Mint を使用する構成の指標とするために測定する．OS node0 で評価プログラムを動作させる．

(構成 3) Mint の OS node0 と OS node1 *(Mint0,1)*

Mint において OS node0 と OS node1 を動作させる構成である．この構成を測定することで，仮想化を使用せずに 1 台の計算機上で複数の OS が動作した場合の性能を明らかにする．OS node0 と OS node1 で同時に評価プログラムを動作させる．

(構成 4) Vanilla Linux 上の単一ゲスト OS *(Linux/Guest1)*

Vanilla Linux において KVM を使用し，ゲスト OS を 1 つ動作させる構成である．この構成は，KVM を使用する構成の指標とするために測定する．GuestOS1 上でのみ評価プログラムを動作させる．

(構成 5) Mint の OS node0 上の単一ゲスト OS *(Mint0/Guest1)*

Mint において OS node0 のみを動作させ，OS node0 で KVM を使用し，ゲスト OS1 を動作させる構成である．この構成を測定することで，Mint において KVM を使用した際の性能を明らかにする．GuestOS1 上で評価プログラムを動作させる．

(構成 6) MintOS における複数 KVM *(Mint0/Guest1,Mint1/Guest2)*

Mint の OS node0 と OS node1 で KVM をそれぞれ使用し，各 OS node でゲスト OS を 1 つを動作させる構成である．この構成を測定することで，1 台の計算機上で複数の KVM が動作した際の性能を明らかにする．GuestOS1 と GuestOS2 で同時に評価プログラムを動作させる．

(構成 7) Vanilla Linux 上の複数ゲスト OS *(Linux/Guest1,2)*

Vanilla Linux において KVM を使用し，ゲスト OS を 2 つ動作させる構成である．この構成を測定することで，KVM 上で複数の OS が動作した際の性能を明らかにする．GuestOS1 と GuestOS2 で同時に評価プログラムを動作させる．

4.3 評価計算機の環境

本評価で使用した計算機の環境を**表 1** に示し，4.2 節で述べた各構成の資源分配について**表 2** に示し，説明する．なお，KVM において，ゲスト OS が実メモリとして認識するのは，VM の仮想メモリの一部である．このため，ゲ

表 1　評価計算機

メモリ	2GB
CPU	Intel(R) Core(TM) i7
HDD1	ST80815AS(80G，7200rpm)
HDD2	ST80815AS(80G，7200rpm)
Kernel	Linux 3.0.8
QEMU	0.13.0

表 2　各構成の資源配分

構成番号	ホスト OS メモリ，HDD，コア数	ゲスト OS メモリ，HDD，コア数
構成 1	Vanilla Linux 512MB，HDD1，1	—
構成 2	OS node0 512MB，HDD1，1	—
構成 3	OS node0 512MB，HDD1，1	—
	OS node1 512MB，HDD2，1	—
構成 4	Vanilla Linux 512MB，HDD1，1	Vanilla Linux 512MB，HDD1，1
構成 5	OS node0 512MB，HDD1，1	Vanilla Linux 512MB，HDD1，1
構成 6	OS node0 512MB，HDD1，1	Vanilla Linux 512MB，HDD1，1
	OS node1 512MB，HDD2，1	Vanilla Linux 512MB，HDD2，1
構成 7	Vanilla Linux 1024MB，HDD1，2	Vanilla Linux 512MB，HDD1，1
		Vanilla Linux 512MB，HDD2，1

スト OS のメモリは，ゲスト OS が実メモリとして認識している領域の大きさを記述している．

ホスト OS の Linux カーネルのバージョンは 3.0.8 であり，ゲスト OS の Linux カーネルのバージョンは 2.6.35 である．また，HDD1 と HDD2 は，各 OS のカーネルを格納している磁気ディスクのことである．

4.4　評価プログラムの処理内容

評価プログラムを作成し，これを用いて Mint における KVM の性能について評価を行った．評価プログラムの処理の流れについて**図 4** に示し，各評価項目と共にその処理内容を以下で説明する．

(1) I/O 処理
　システムコール `lseek()` と `read()` を用いた磁気ディスクへの I/O 処理 (繰り返し回数:L 回) を行う．なお，`lseek()` を発行する際は，ランダムな位置にヘッドを移動させてからデータを読み込む．

(2) レジスタ操作処理
　レジスタ値をインクリメントする処理 (繰り返し回数:M 回) を行う．

(3) データ複写処理
　システムコール `memcpy()` を用いてメモリを複写する処理 (繰り返し N 回) を行う．

この評価プログラムは，I/O 処理，レジスタ操作処理，およびデータ複写処理の繰り返し回数 L，M，N を固定し，

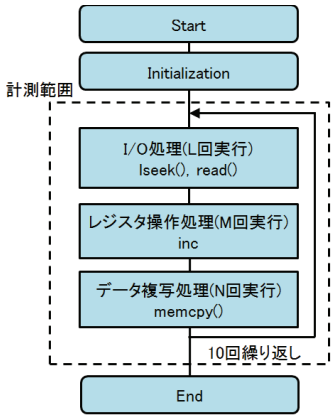

図 4　評価プログラムの処理内容

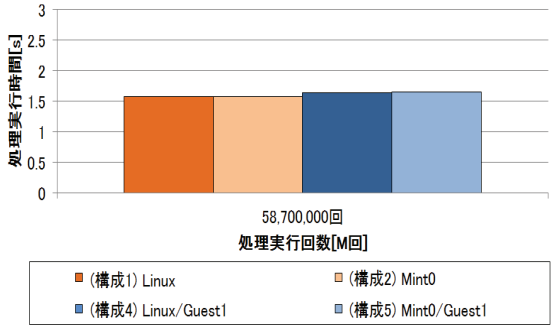

図 5　個別処理実行時間の比較 (レジスタ操作)

繰り返し 10 回実行するまでの処理実行時間を測定する．処理実行時間の測定には TSC(Time Stamp Counter) レジスタの値を用いる．

5.　基本性能の評価結果と考察

5.1　Mint の基本性能

(構成 1)*Linux*，(構成 2)*Mint0*，(構成 4)*Linux/Guest1* および (構成 5)*Mint0/Guest1* において，4.4 節で述べた評価プログラムを用いて測定したレジスタ操作処理，データ複写処理，および I/O 処理の評価結果について考察する．なお，データ複写処理と I/O 処理において，読み込むデータサイズは 256KB，512KB，1024KB とした．レジスタ操作処理の評価結果を**図 5**，データ複写処理の評価結果を**図 6**，I/O 処理の評価結果を**図 7** にそれぞれ示す．

各図から，Vanilla Linux と Mint の性能は同等であるといえる．具体的には，レジスタ操作処理，データ複写処理，および I/O 処理の全ての評価において，(構成 1)*Linux* と (構成 2)*Mint0* の処理実行時間に差がない．同様に，(構成 4)*Linux/Guest1* と (構成 5)*Mint0/Guest1* の処理実行時間に差がない．よって，計算機上で評価プログラムを 1 つだけ動作させた場合 (以後，個別処理と呼ぶ)，Vanilla Linux と Mint の基本性能は同等であるといえる．

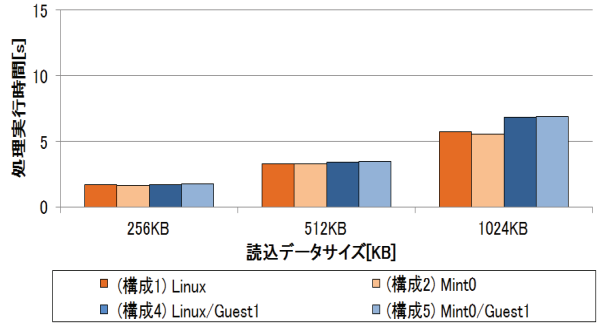

図 6 個別処理実行時間の比較 (データ複写)

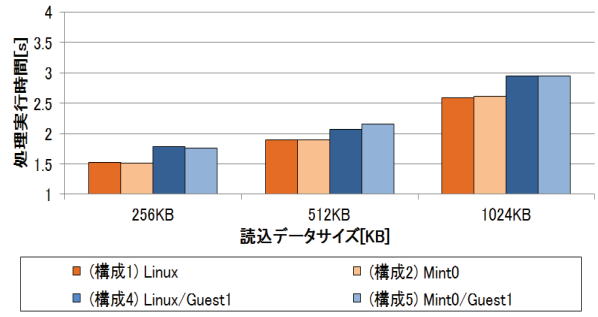

図 7 個別処理実行時間の比較 (I/O)

5.2 Mint で KVM を使用した場合

5.2.1 レジスタ操作処理性能

4.4 節で説明した評価プログラムにおいて，$L:M:N=0:1:0$ とし，レジスタ操作処理の実行時間を測定した．この測定結果のうち個別処理を図 5，計算機上で異なる OS に 2 つの評価プログラムを同時に動作させた場合 (以後，同時処理と呼ぶ) を**図 8** に示し，以下で考察する．

(1) レジスタ操作処理性能は Vanilla Linux，Mint，KVM の構成に影響しない．

図 5 と図 8 において，処理実行時間は構成によらず，ほぼ同等である．まず，図 5 について考察する．KVM を使用しない構成 (構成 1)*Linux*, (構成 2)*Mint0* とそれらの上に KVM を動作させる構成 (構成 4)*Linux/Guest1*, (構成 5)*Mint0/Guest1* の処理実行時間が同等である．これは，以下の 2 つが原因であると考えられる．

(a) ホスト OS とゲスト OS に評価以外のプロセスが動作していない．

ホスト OS で動作するプロセスは VM を動作させるプロセスのみである．同様に，ゲスト OS のプロセスは評価プログラムのみである．このため，KVM を使用して動作する VM の仮想 CPU の処理はホスト OS のプロセススケジューラの影響をほとんど受けていないと考えられる．

(b) センシティブ命令を発行していない．

本評価プログラムにおける CPU 評価では，レジスタにインクリメントを行う．このため，KVM 上で動作するゲスト OS は VM Exits 処理と仮想化のオーバーヘッドが発生しない．

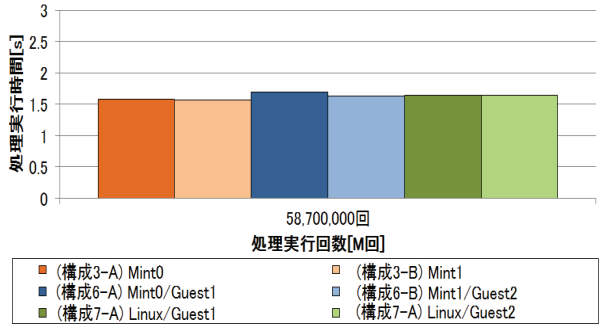

図 8 同時処理実行時間の比較 (レジスタ操作)

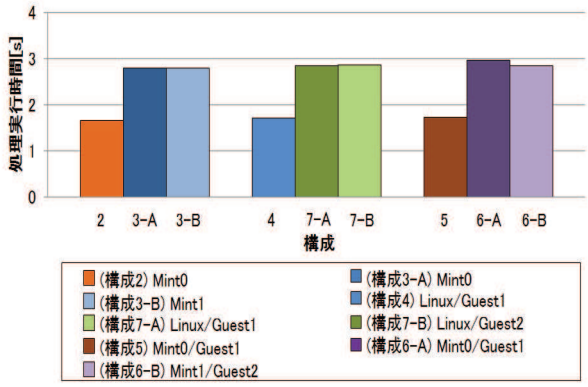

図 9 個別処理と同時処理の比較 (データ複写 [256KB])

次に，図 8 について考察する．(構成 3)*Mint0,1* は，各 OS node に割り当てられた実 CPU が評価プログラムを実行している．(構成 6)*Mint0/Guest1,Mint1/Guest2* は，各 OS node に割り当てられた実 CPU が，VM が持つ仮想 CPU の処理を実行している．また，各 OS node のプロセスは，それぞれ 1 台の VM が動作するためのプロセスのみであり，各 VM のプロセスは評価プログラムのみである．このため，実 CPU が VM 上で動作する評価プログラムを実行している．(構成 7)*Linux/Guest1,2* は，Vanilla Linux に割り当てられた 2 つの実 CPU コアが，それぞれ 1 台の VM が持つ仮想 CPU の処理をホスト OS のスケジューラによって割り当てられ，実行している．したがって，ホスト OS のプロセスとホスト OS が占有するコア数が同じであれば，レジスタ操作処理性能は変化しない．

5.2.2 データ複写処理性能

4.4 節で説明した評価プログラムにおいて，$L:M:N=0:0:1$ とし，読み込むデータサイズを 256KB，512KB，1024KB としてデータ複写処理の実行時間を測定した．この測定結果のうち，個別処理の結果を図 6 示す．また，データサイズを 256KB にした場合の個別処理と同時処理を比較した結果を**図 9** に示し，以下で考察する．

(1) KVM を使用するとデータ複写処理性能が低下する．

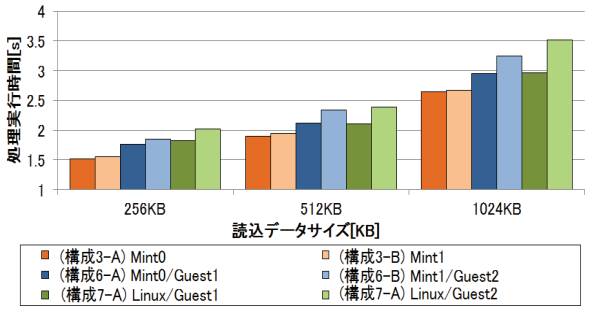

図 10 同時処理実行時間の比較 (I/O)

図 6 の 1024KB のデータを読み込む場合において，KVM を使用しない構成 (構成 1)*Linux*, (構成 2)*Mint0* とそれらの上に KVM を動作させる構成 (構成 4)*Linux/Guest1*, (構成 5)*Mint0/Guest1* の処理実行時間を比較すると (構成 4)*Linux/Guest1*, (構成 5)*Mint0/Guest1* は共に約 20 %増加している．KVM において，ゲスト OS が物理メモリとして認識するのはホスト OS の仮想メモリの一部である．このため，KVM を使用した場合，ゲスト OS の仮想アドレスから 1024KB のマッピングを行う際に何らかの要因で処理実行時間が長くなると考えられる．

(2) データ複写処理が同時に発生すると処理性能が低下する．

図 9 において，個別処理と同時処理を比較するとデータ複写処理性能が低下している．具体的には，Mint において，KVM を使用していない構成 (構成 2)*Mint0* と (構成 3)*Mint0,1* の処理実行時間を比較すると (構成 3)*Mint0,1* は約 70 %増加している．また，Vanilla Linux において KVM を使用する構成 (構成 4)*Linux/Guest1* と (構成 7)*Linux/Guest1,2* の処理実行時間を比較すると，(構成 7)*Linux/Guest1,2* は約 67 %増加している．さらに，Mint において KVM を使用する構成 (構成 5)*Mint0/Guest1* と (構成 6)*Mint0/Guest1,Mint1/Guest2* の処理実行時間を比較すると (構成 6)*Mint0/Guest1,Mint1/Guest2* は約 71 %増加している．なお，これは読み込むデータサイズを 512KB，1024KB に変更した際も同様に増加している．これらは，メモリアクセスにともなうバスの競合が原因であると考えられる．

5.2.3 I/O 処理性能

4.4 節で説明した評価プログラムにおいて，$L : M : N = 1 : 0 : 0$ とし，読み込むデータサイズを 256KB，512KB，1024KB として I/O 処理の実行時間を測定した．この測定結果を図 7，**図 10** に示し，以下で考察する．

(1) KVM を使用すると I/O 処理性能が低下する．

図 7 において，KVM を使用しない構成 (構成 1)*Linux*, (構成 2)*Mint0* とそれらの上に KVM を動作させる構成 (構成 4)*Linux/Guest1*, (構成 5)*Mint0/Guest1* の処理実行時間を比較すると (構成 4)*Linux/Guest1*, (構成 5)*Mint0/Guest1* は共に約 16 %増加している．これは，KVM では QEMU がゲスト OS の命令を物理デバイスが実行できるように I/O 命令を変更する処理が発生するためである．この処理負荷が存在するため，KVM を使用した場合の処理実行時間が長くなる．

(2) 計算機資源と VM の数が同じであれば，複数のハイパーバイザに分散させた方が処理が速くなる

図 10 における (構成 6)*Mint0/Guest1,Mint1/Guest2* と (構成 7)*Linux/Guest1,2* の処理実行時間を比較すると，(構成 7)*Linux/Guest1,2* の処理実行時間が約 8 %長い．(構成 6)*Mint0/Guest1,Mint1/Guest2* は OS node0 上で GuestOS1 が動作し，OS node1 上で GuestOS2 が動作し，2 台のゲスト OS が同時に評価プログラムを実行している．ゲスト OS の処理は OS node ごとに 1 つの CPU に割り当てられ，デバイスドライバに命令を通知する．一方，(構成 7)*Linux/Guest1,2* では Vanilla Linux 上で GuestOS1 と GuestOS2 が同時に動作し，2 台のゲスト OS が同時に評価プログラムを実行している．このため，GuestOS1 の処理と GuestOS2 の処理は，ホスト OS のスケジューリングによってそれぞれ 1 つのコアが割り当てられ，デバイスドライバに命令を通知する．この際，(構成 6)*Mint0/Guest1,Mint1/Guest2* と (構成 7)*Linux/Guest1,2* において，処理が異なっている．(構成 6)*Mint0/Guest1,Mint1/Guest2* は各 OS node が磁気ディスクを 1 つずつ占有して動作している．このため，(構成 6)*Mint0/Guest1,Mint1/Guest2* の各 OS node がロードしているデバイスドライバは占有している磁気ディスクしか登録されていない．これに対し，(構成 7)*Linux/Guest1,2* おけるデバイスドライバには両方の磁気ディスクが登録されている．よって，磁気ディスクからメモリにデータを転送する際の処理が異なっており，処理実行時間に違いがあると考えられる．

6. おわりに

Mint を利用して複数ハイパーバイザを走行させる方式について述べた．具体的には，Mint について説明し，その制約について述べた．そして，Mint を利用して複数ハイパーバイザを動作させる方式について，目的，および Mint で KVM を使用した際の構成について述べた．

次に，Mint で KVM を動作させ，レジスタ操作処理，データ複写処理，および I/O 処理の基本評価結果を述べた．この評価の目的は 2 つある．1 つは，Mint 上の KVM と通常の Linux 上の KVM の性能が同等であることを明らかにすることである．もう 1 つは，Mint で複数の KVM を使用し，VM を動作させた場合の処理性能について明らか

にすることである．まず，I/O 処理，レジスタ操作処理，データ複写処理の性能は Mint 上の KVM と通常の Linux 上の KVM の性能が同等であることを示した．また，I/O 処理については複数の VM を用意し，同時に評価プログラムを実行した場合，KVM を複数動作し，分散させることで処理負荷の影響を限定できることについて示した．なお，レジスタ操作処理とデータ複写処理については，KVM を複数動作し，VM を分散させて動作させた場合と通常の Linux で KVM を使用した場合の処理性能が同等であることを示した．

謝辞 本研究の一部は, 科学研究費補助金基盤研究 (B)(課題番号:24300008) による．

参考文献

[1] Paul, B., Boris, D., Keir, F., Steven, H., Tim, H., Alex, H., Rolf, N., Ian, P. and Andrew, W. :Xen and the Art of Virtualization, *Proc. of the 19th ACM Symposium on Operating Systems Principles*, pp.164-177, 2003.

[2] Avi, K. Yaniv, K. Dor, L. Uri, L. and Anthony, L. :kvm: the Linux Virtual Machine Monitor, *Proc of the Linux Symposium, Ottawa, Ontario*, pp. 225-230 (2007).

[3] Jeremy, S. Ganesh, V. and Beng-Hong, L. : Virtualizing I/O De-vices on VMware Workstation's Hosted Virtual Machine Monitor, *Proc. of the GeneralTrack: 2002 USENIX Annual Technical Conference*, pp.1-14, 2001.

[4] Hitoshi, U. and Satomi, H. :Virtage: Hitachi's Virtualization Technology, *Proc GPC '09 Proceedings of the 2009 Workshops at the Grid and Pervasive Computing Conference*, pp.121-127, (2009).

[5] 千崎良太，中原大貴，牛尾裕，片岡哲也，栗田祐一，乃村能成，谷口秀夫，: マルチコアにおいて複数の Linux カーネルを走行させる Mint オペレーティングシステムの設計と評価, 電子情報通信学会技術研究報告, vol.110, no.278, pp.29-34, (2010.11).

[6] 中原 大貴, 乃村 能成, 谷口 秀夫,: 32/64bit カーネル混載方式の実現 , 電子情報通信学会技術研究報告, vol.111, no.255, pp.25-30 (2011.10).

[7] Hariprasad Nellitheertha, : Reboot Linux faster using kexec, IBM (online), available from 〈https://www.ibm.com/developerworks/linux/library/l-kexec.html〉 (accessed 2012-02-07)