特 別 研 究 報 告 書

題 目

Mintオペレーティングシステム上のKVMの評価

指導教員

報 告 者

仲尾 和祥

岡山大学工学部 情報工学科

平成25年2月8日 提出

要約

1台の計算機上で多様なサービスを提供する手段として，1台の計算機上に複数のオペレーティングシステムが走行させる方式が利用されている．この方式の1つとして仮想計算機方式がある．仮想計算機方式は，OSが走行できる環境を提供するソフトウェアであるハイパーバイザを利用することで，複数のOSを走行させる方式である．この方式では，ハイパーバイザ上で走行させる各OSに資源を共有し，柔軟に利用できる利点がある．しかし，OS間には処理負荷の影響があるほか，ハイパーバイザが単一障害点となる問題がある．

一方，私の所属する研究グループでは，1台のマルチコア計算機上で複数Linuxを独立に走行させる方式のOSとして，Mint(Multiple Independent operating systems with New Technology)オペレーティングシステムを開発している．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ほか，カーネルに変更を加える必要がある．

そこで，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排除しつつ，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範囲を限定するほか，Mintで未改修のカーネルを走行可能にする．そして，基本評価により，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されることを示した．具体的には，I/O処理の処理負荷の影響範囲を限定でき，CPU処理とメモリ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．

目次

図 目 次

表 目 次

第 1 章

はじめに

計算機の性能が向上している．このため，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] や KVM[2] といった仮想計算機方式 (以後，VM 方式と呼ぶ) がある．

VM 方式は，ハイパーバイザを使用し，複数の OS を走行させる方式である．この VM 方式では，ハイパーバイザ上で走行させる各 OS の資源を共有し，柔軟に利用できる利点がある．しかし，OS 間には処理負荷の影響があるほか，ハイパーバイザが単一障害点となる問題がある．この問題を解決するため，CPU とメモリを論理分割し，デバイスを仮想化して複数の OS を動作させる Virtage[3] 上に複数の KVM を動作させる方式がある [4]．しかし，この方式では次の 3 つの問題がある．具体的には，Virtage が実行できる環境は特定のハードウェアでのみ提供されていること，デバイスの仮想化を行うためのオーバーヘッドが発生すること，および Virtage が単一障害点となることである．

一方で，私の所属する研究グループでは，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(Multiple Independent operating systems with New Technology) 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．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ほか，カーネルに変更を加える必要がある．

そこで，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．これにより，ハイパーバイザ間の処理負荷の影響を排除しつつ，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範囲を限定するほか，Mint で未改修のカーネルを走行可能にする．

本論文では，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，評価する．2 章では VM 方式とその問題点について説明する．3 章では Mint について説明し，その制約について述べる．4 章では 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，利用するハイパーバイザ，および構成について述べる．5 章では提案方式の基本評価を行い，6 章で評価結果と考察について述べる．

第 2 章

仮想計算機方式

2.1　仮想計算機方式の構造

VM 方式とは，ハイパーバイザを使用し，複数の OS を走行させる方式である．なお，ハイパーバイザとは複数の OS が走行できる環境を提供するソフトウェアであり，各 OS が走行できる環境を仮想計算機 (以後，VM と呼ぶ) と呼ぶ．これにより，1 台の計算機上で複数の OS を走行可能にしている．VM 方式により，VM を動作させる際の構造について図 2.1 に示し，以下で説明する．

(1) ベアメタルハイパーバイザ方式
ハイパーバイザ方式は Xen や VMware vSphere で用いられる方式である．この方式による仮想化では，ハードウェア上に計算機の資源管理と VM の制御を行うプログラムとしてハイパーバイザを動作させ，このハイパーバイザ上で VM を動作させる．この際，動作させている VM の命令はハイパーバイザを介してハードウェアが直接実行できる．このため，仮想化を行う際のオーバーヘッドは小さいという利点がある．

(2) ホスト OS 方式
ホスト OS 方式は KVM や VMware Workstation[6] で用いられる方式である．この方式による仮想化では，ハードウェア上で OS を動作させ，この OS のプロセスとしてハイパーバイザを動作させ，ハイパーバイザ上で VM を動作させる．なお，ハードウェア上で動作させる OS をホスト OS と呼び，VM 上で動作させる OS をゲスト OS と呼ぶ．ホスト OS 方式では，ハイパーバイザがゲスト OS から発行された命令を解析し，ホスト OS がゲスト OS からの命令をプロセスとして処理する．このため，仮想化を行う際のオーバーヘッドは大きくなる欠点がある．

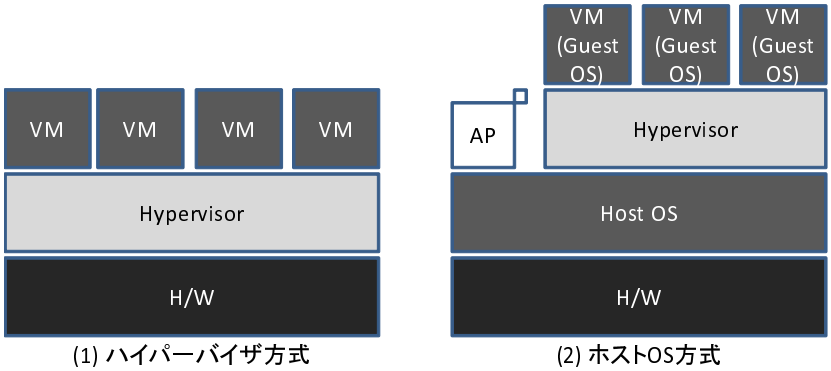


図　2.1　仮想計算機の構成例

2.2　仮想計算機方式の問題点

VM 方式の問題として以下の 2 つが挙げられる．

(1) ハイパーバイザに不具合が生じると全ての VM が停止する．
VM の異常やハイパーバイザ自身の不具合によりハイパーバイザが停止すると，計算機上の全ての VM が停止する．

(2) VM 間の影響が存在する．
VM 方式では，ハイパーバイザの管理の下で，全ての VM が資源を共有している．このため，1 つの VM の処理が高負荷になると，ハイパーバイザがこの VM に資源を大きく割り当て，他の VM の動作が遅くなる．このように，同じハイパーバイザ上で動作する全ての VM は相互に処理負荷の影響を与えてしまう．

第 3 章

Mint オペレーティングシステム

3.1 Mint の設計方針

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．

(2) 全ての Linux が入出力性能を十分に利用できる．

3.2 Mint の構成

Mint の構成例を図 3.1 に示し，Mint における CPU，メモリ，および入出力機器の分割方法について以下で説明する．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．

(1) CPU
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．

(2) メモリ
実メモリ領域を空間分割し，各 OS node に分配する．

(3) 入出力デバイス
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．

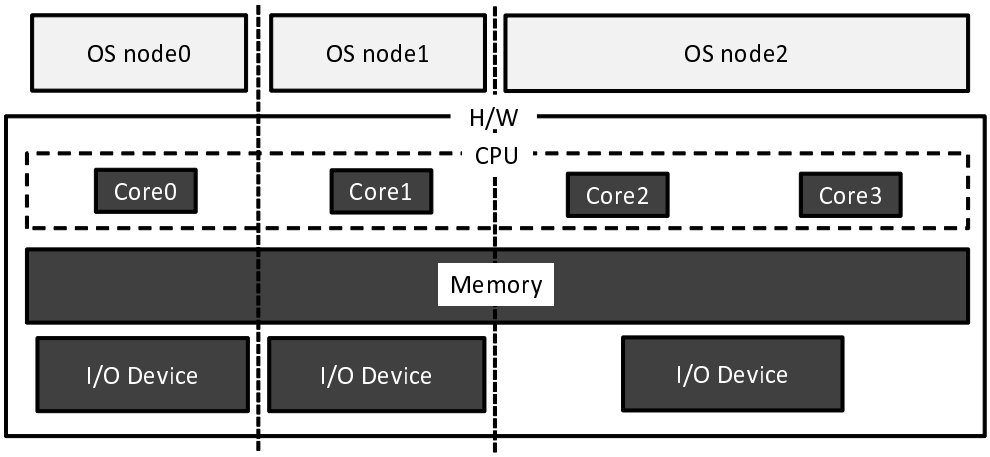


図 3.1 Mint の構成例

3.3 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 Linux を走行できない．また，走行する Linux の数だけ磁気ディスクを用意し，各 Linux 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 のカーネルを変更する必要がある．
Mint では，複数 Linux のハードウェア上で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 を走行させることができない．

第 4 章

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

4.1 目的

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 4.1 に示し，以下で目的を 4 つ述べる．

(1) ハイパーバイザが異常終了した際の影響範囲を限定する．
VM 方式において，2.2 節の (1) で述べたように，ハイパーバイザに不具合が生じると全ての VM が停止する問題がある．そこで，1 台の計算機上で複数のハイパーバイザを動作させ，各ハイパーバイザ上に VM を分散させることにより，ハイパーバイザが異常終了した際の VM への影響を限定する．例えば，図 4.1 の構成において，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 は問題なく動作可能である．

(2) VM の処理負荷による影響範囲を限定する．
VM 方式において，2.2 節の (2) で述べたように，全ての VM 間で処理負荷の影響が発生する問題がある．この問題は，ハイパーバイザ上で動作する VM に発生する問題である．このため，複数ハイパーバイザを動作させ，VM 間の影響の範囲をハイパーバイザごとに分割することで，VM の処理負荷による影響をハイパーバイザごとに限定する．この例を図 4.1 を用いて説明する．Mint 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発生しない．

(3) 動作する計算機の資源において，共有する箇所と共有しない箇所を作成する．
Mint において，3.3 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することができる．この例を図 4.1 を用いて説明する．2 枚の NIC を搭載している計算機において，Mint として NIC を 1 枚ごとに分割して占有した OS node1 と OS node2 を動作させる．この際，OS node1 と OS node2 がそれぞれ複数の VM を動作させ，OS node1 上の VM は OS node1 の占有する NIC を共有して動作できる．また，OS node2 の VM は OS node2 の占有する NIC を共有して動作できる．

(4) Mint を構成する計算機上で未改修の OS を動作させる．
Mint において，3.3 節の (2) で述べたように，走行させる OS のカーネルを変更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 において未改修の OS を動作可能にする．この例を図 4.1 を用いて説明する．Mint 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．

4.2 Mint で利用するハイパーバイザ

4.2.1 利用するハイパーバイザ

2.1 節にて，VM 方式はハイパーバイザ方式とホスト OS 方式の 2 つに分けられると述べた．この内，ハイパーバイザ方式はハードウェア上で直接ハイパーバイザを動作させるため，Mint で動作させるハイパーバイザとしては適さない．これは，計算機上で複数のハイパーバイザを動作させるためには，ハイパーバイザのプログラムを大きく変更する必要があると考えられるためである．一方，ホスト OS 方式はハードウェアの間にホスト OS を動作させ，ゲスト OS とハイパーバイザをホスト OS のプロセスとして動作させる．また，Mint で起動した Linux は通常の Linux として利用できる．このため，Linux をホスト OS として動作させるハイパーバイザを Mint で利用できると考えられる．

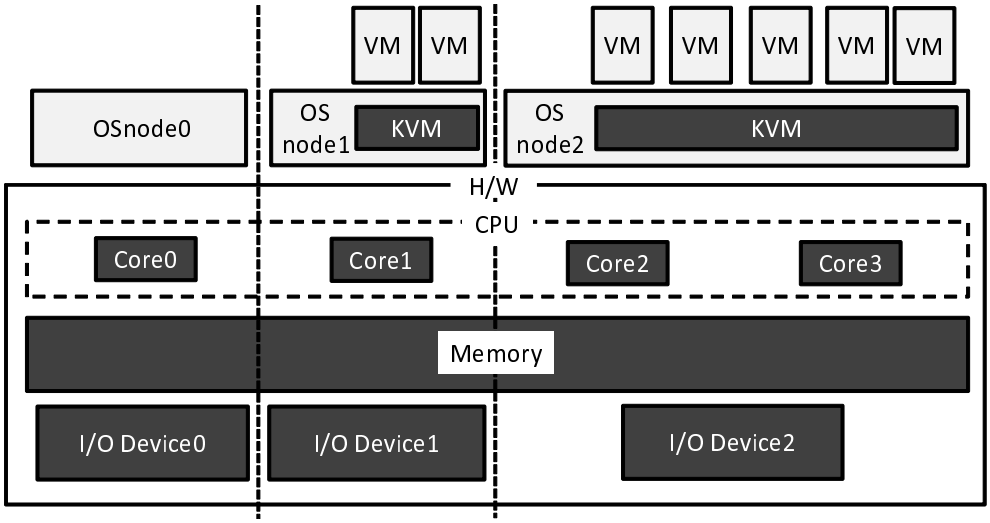


図　4.1　Mint において複数 VM の動作させた構成例

Linux をホスト OS とするハイパーバイザとして，KVM と VMware Workstation が広く利用されている．このうち，KVM の特徴を以下に示す．

(1) Linux に組み込まれたハイパーバイザである．
KVM は Linux に組み込まれたハイパーバイザであり，CPU の仮想化支援機能を利用して実装されている．また，Linux 自体をホスト OS としてゲスト OS の制御を行うことができる．

(2) 各 VM はホスト OS のユーザプロセスとして動作する．
各 VM は，ホスト OS における単一のユーザプロセスとして動作する．なお，VM からの資源の要求は KVM を経由して行われている．また，ホスト OS は通常のプロセスの処理と同様にゲスト OS の処理を実行するため，通常の Linux として使用することができる．

上記の特徴から，KVM は Mint 上で問題なく動作できる．このため，Mint で利用するハイパーバイザの例として，KVM を利用する．

4.2.2　KVM(Kernel-based Virtual Machine)

KVM は，デバイスエミュレーションを行う QEMU と連携することで VM を動作させる．KVM を用いて VM を起動した際の構成例を図 4.2 に示し，以下で KVM と QEMU の役割を説明する．

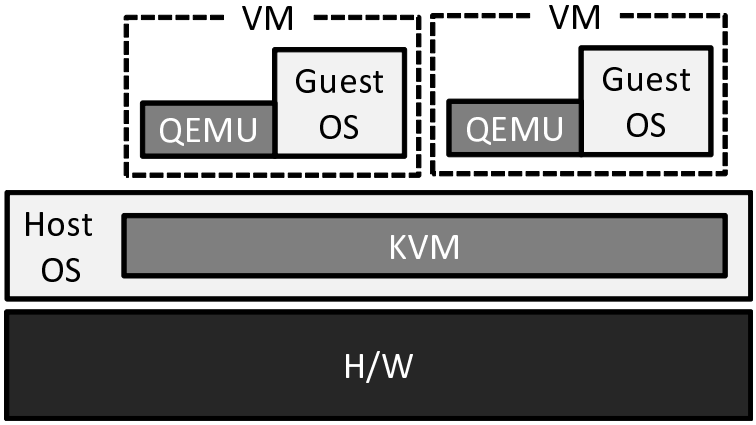


図 4.2 KVM の構成例

(1) QEMU

(A) ホスト OS へのシステムコールの発行
ホスト OS のデバイスを使用するためのシステムコールを発行する．具体的には，VM の起動処理とゲスト OS が発行した I/O 命令をデバイスドライバに発行するため，システムコールを使用してホスト OS に命令を発行する．

(B) ハードウェアのエミュレート
VM のハードウェアをエミュレートし，ゲスト OS が発行した I/O 命令をホスト OS のデバイスが使用できるように変更する．また，エミュレートを行う VM のハードウェアの起動，停止，およびマイグレーションを行う．

(2) KVM

(A) CPU の動作モードの変更
ゲスト OS の命令に応じて CPU の仮想化支援機能を利用して CPU の動作モードを変更する．

(B) CPU から通知される例外処理の判断
CPU の仮想化支援機能がゲスト OS の命令に応じて KVM に例外処理であることを通知する．KVM は通知された内容に応じて，ゲスト OS の処理に戻るのか QEMU がデバイスのエミュレートをする必要があるのか判断する．

(C) VM の作成
QEMU から発行されたシステムコールに応じて新たに VM を作成する．

4.3 構成

Mint においてハイパーバイザを使用し，複数の VM を動作させる際の構成例として図 4.1 について説明する．

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 はそれぞれハードウェア上で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 が占有している入出力デバイス 1 は，OS node1 上で動作している VM 間で共有して利用することができ，OS node2 が占有している入出力デバイス 2 は，OS node2 上で動作している VM 間で共有して利用することができる．

さらに，構成例では，KVM を用いて VM を動作させているため，VM 上で動作する OS のカーネルを変更せずに動作できる．

第 5 章

評価

5.1 評価の目的

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 Linux で KVM を使用した場合の性能の比較を行う.

5.2 評価構成

5.3 節で示した各 OS を組み合わせた構成で比較評価を行う．評価を行う構成を図 5.1 を示し，説明する.

(構成 1) Vanilla Linux (*Vanilla Linux1*)

Vanilla Linux1 を動作させる構成である．Vanilla Linux で評価プログラムを動作させる.

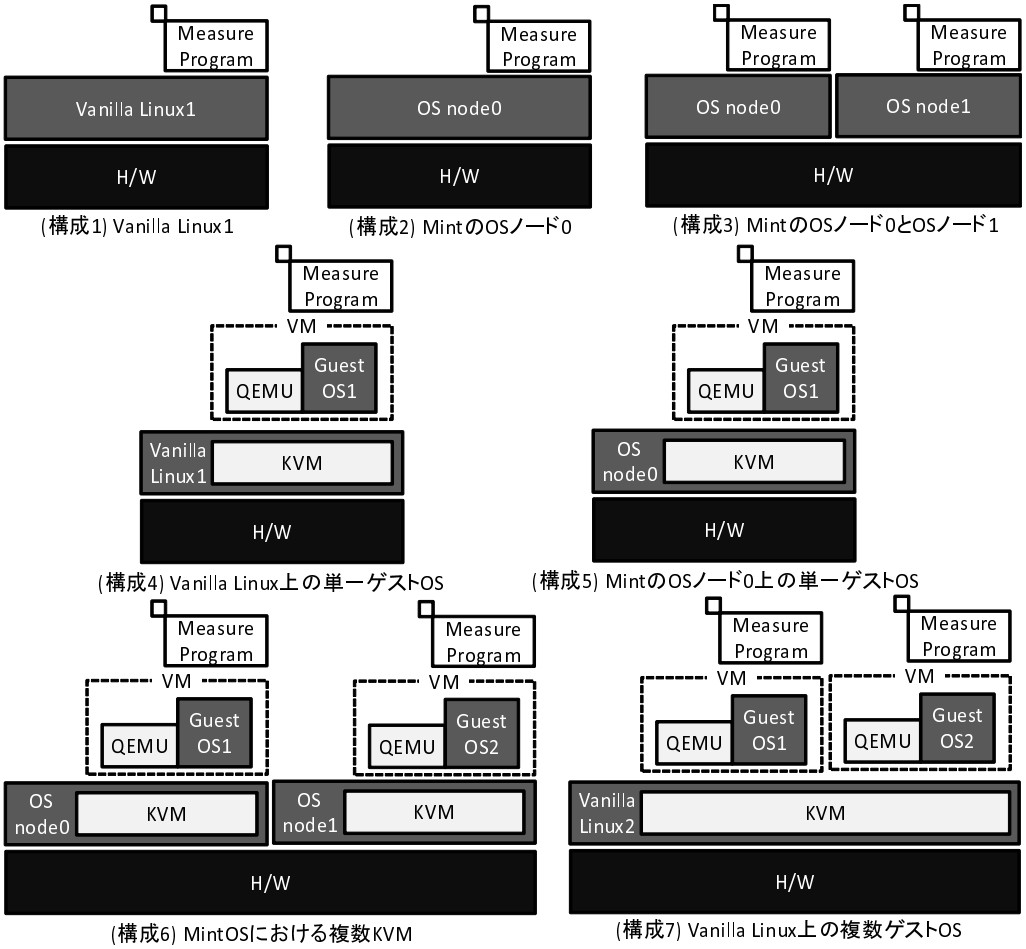


図　5.1　評価を行う際の構成図

(構成 2) Mint の OS ノード 0 (*OS node0*)

Mint において OS ノード 0 のみを動作させる構成である．OS ノード 0 で評価プログラムを動作させる．

(構成 3) Mint の OS ノード 0 と OS ノード 1 (*OS node0,1*)

Mint において OS ノード 0 と OS ノード 1 を動作させる構成である．OS ノード 0 と OS ノード 1 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 (*Vanilla Linux/KVM/GuestOS1*)

Vanilla Linux1 において KVM を使用し，ゲスト OS1 を動作させる構成である．ゲスト OS1 上でのみ評価プログラムを動作させる．

(構成 5) Mint の OS ノード 0 上の単一ゲスト OS (*OS node0/KVM/GuestOS1*)

Mint において OS ノード 0 のみを動作させ，OS ノード 0 で KVM を使用し，ゲスト OS1 を動作させる構成である．ゲスト OS1 上で評価プログラムを動作させる．

(構成 6) Mint における複数 KVM (*OS node0/KVM/GuestOS1,OS node1/KVM/GuestOS2*)
Mint の OS ノード 0 と OS ノード 1 で KVM をそれぞれ使用し，OS ノード 0 ではゲスト OS1，OS ノード 1 ではゲスト OS2 を動作させる構成である．ゲスト OS1 とゲスト OS2 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 (*Vanilla Linux2/KVM/GuestOS1,2*)
Vanilla Linux2 において KVM を使用し，2 つのゲスト OS を動作させる構成である．ゲスト OS1 とゲスト OS2 で同時に評価プログラムを動作させる．

5.3 比較対象 OS の環境

本評価で使用した OS の環境を表 5.1 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 Linux Kernel 3.0.8 であり，ゲスト OS の Linux カーネルのバージョンは Linux Kernel 2.6.35 である．なお，本評価では KVM を使用するため，全ての OS はカーネルを構築する際に以下の 2 つのモジュールを追加する．

(1) Kernel-based Virtual Machine (KVM) support
KVM を使用するためのモジュール

(2) KVM for Intel processors support
Intel CPU がサポートする VT-x 機能を使用可能にするモジュール

また，HDD1 と HDD2 は，それぞれ Mint において OS ノード 0 と OS ノード 1 がそれぞれ占有する磁気ディスクのことである．

5.4 基本評価

5.4.1 評価の概要

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う．この評価プログラムの評価項目について以下で説明する．

表 5.1 評価環境

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 Linux1 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 Linux1 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 Linux2 1024MB，HDD1，2	Vanilla Linux 512MB，HDD1，1
		Vanilla Linux 512MB，HDD1，1

(1) I/O 処理
各 OS において，磁気ディスクへアクセスした後，ブロック単位の読み込み処理の性能を測定する．

(2) CPU 処理
各 OS のプロセッサ処理の性能を測定する．

(3) メモリ処理
各 OS がメモリへアクセスする処理の性能を測定する．

5.4.2 評価プログラムの処理内容

評価プログラムの処理の流れについて図 5.2 に示し，各評価項目の処理の内容を以下で説明する．

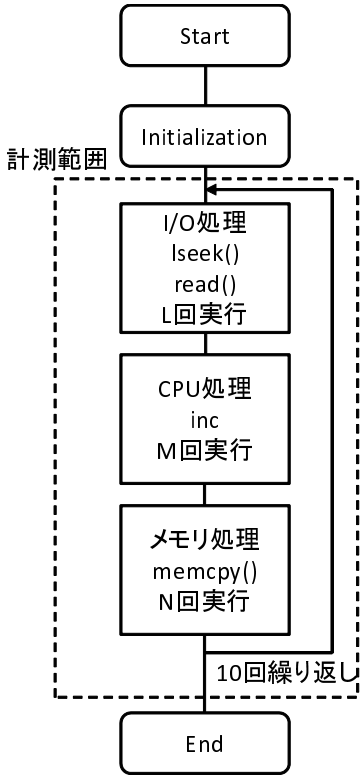


図 5.2 評価プログラムの処理内容

(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，ランダムな値を用いてヘッダを移動させることにより，ランダムな位置からデータを読み込む．これは，磁気ディスクがキャッシュデータを読み込むことによる測定結果への影響を減らすためである．

(2) CPU 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) メモリ処理

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．

また，この評価プログラムは，I/O 処理，CPU 処理，およびメモリ処理の繰り返し回数 L，M，N を固定し，繰り返し 10 回実行するまでの処理実行時間を測定する．なお，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．TSC レジスタとは，計算機の電源投入時からクロックサイクル数を参照するためのレジスタである．

第 6 章

結果と考察

6.1　基本性能の評価

6.1.1　CPU 処理

5.4.2 項で説明した評価プログラムにおいて，$L : M : N = 0 : 1 : 0$ とし，CPU 処理の実行時間を測定した．この測定結果を図 6.1 に示し，以下で考察する．

(1) 個別に処理を行った場合

(A) KVM の使用時と不使用時は同等の性能である．

図 6.1(1) において，KVM を使用しない構成 (構成 1)，(構成 2) と比べて，それらの上に KVM を動作させる構成 (構成 4)，(構成 5) の処理実行時間が同等である．これは，KVM では，VM の仮想 CPU 処理を行うまでのプロセスはハードウェアで実行されている．具体的には，ゲスト OS が仮想 CPU のレジスタにインクリメントを行う命令を発行すると，実 CPU の仮想化支援機能が発行された命令を解析し，KVM へ例外を通知し仮想 CPU のレジスタにインクリメントを行う．なお，KVM を使用して動作する VM において，仮想 CPU の処理はホスト OS のプロセススケジューラによって定期的に実 CPU の実行時間が割り当てられている．よって，KVM を使用しても CPU 処理の性能は KVM を使用しない場合と同等である．

(B) Vanilla Linux と Mint の性能は同等である．

図 6.1(1) において，(構成 1) と (構成 2) の処理実行時間に差がない．同様に，(構成 4) と (構成 5) の処理実行時間にも差がない．このため，KVM を使用する場合

と使用しない場合の両方において Vanilla Linux と Mint では CPU 処理の性能は同等である．

(2) 同時に処理を行った場合

(A) 複数の VM を動作させる場合の CPU 処理性能はハイパーバイザの個数で変化しない．

図 6.1(2) において，(構成 6) と (構成 7) の処理実行時間はほぼ同等である．(構成 6) は 1 つの仮想 CPU を 1 つの実 CPU が実行している．(構成 7) は 2 つの仮想 CPU を 2 つの実 CPU が実行している．この際，処理実行時間に差が存在しない．よって，ホスト OS のコア数と，仮想 CPU が同数であれば，1 つのハイパーバイザ上で複数の VM を動作させる場合と複数のハイパーバイザ上で複数の VM を動作させる場合の CPU 処理は同等である．

6.1.2 メモリ処理

5.4.2 項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 としてメモリ処理の実行時間を測定した．この測定結果を図 6.2 に示し，以下で考察する．

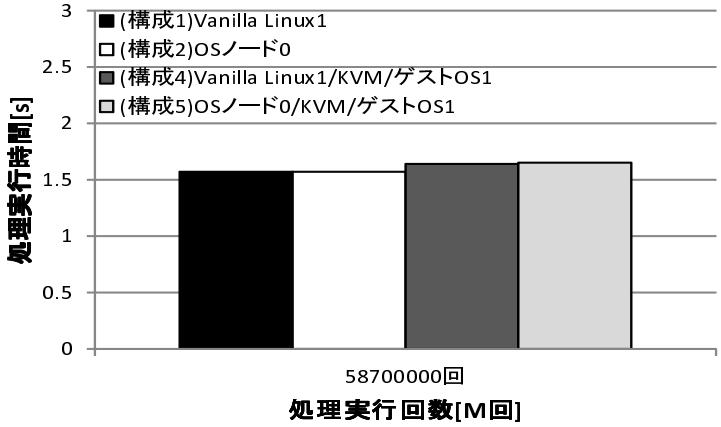
(1) 個別に処理を行った場合

(A) KVM を使用すると性能が低下する．

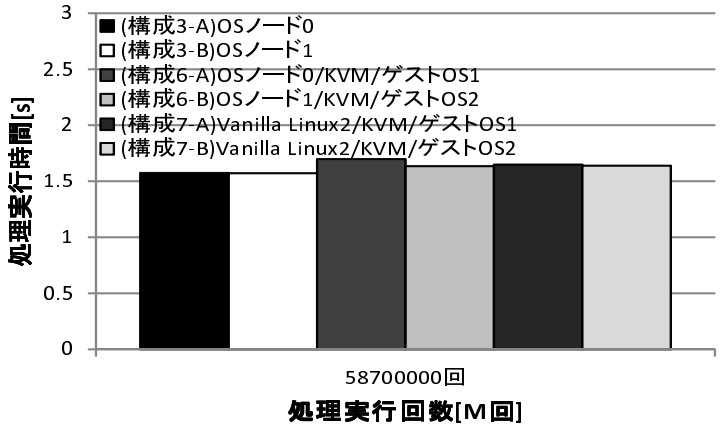
図 6.2(1) の 1024KB のデータを読み込む場合において，KVM を使用しない構成 (構成 1)，(構成 2) と比べて，それらの上に KVM を動作させる構成 (構成 4)，(構成 5) の処理実行時間が長い．具体的には，KVM を使用しない構成 (構成 1)，(構成 2) と比べて，それらの上に KVM を動作させる構成 (構成 4)，(構成 5) の処理実行時間は共に約 20 %増加している．KVM において，ゲスト OS が物理メモリとして認識するのはホスト OS の仮想メモリの一部である．このため，ゲスト OS の仮想メモリは，ホスト OS の仮想メモリ，ホスト OS の物理メモリの順でマッピングが行われる．よって，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．

(B) Vanilla Linux と Mint の性能は同等である．

図 6.1(1) において，(構成 1) と (構成 2) の処理実行時間に差がない．同様に，(構



(1) 個別処理



(2) 同時処理

図　6.1　処理実行時間の比較 (CPU)

成 4) と (構成 5) の処理実行時間にも差がない．よって，個別に処理を行った場合，Vanilla Linux と Mint のメモリ処理の性能は同等である．

(2) 同時に処理を行った場合

(A) 複数の VM を動作させる場合のメモリ処理性能はハイパーバイザの個数で変化しない．

図 6.2(2) において，(構成 6) と (構成 7) と比較した際，処理実行時間に差がない．(構成 6) は，各ゲスト OS のプロセスは 2 段階のマッピングを行った後，CPU がメモリにアクセスする．ここで，Mint は，実メモリを各 OS で分割して占有している．しかし，2 つのプロセスが 1 つの実メモリに同時にアクセスできない．こ

のため，一方のプロセスが待機することで，個別処理に比べて処理実行時間が長くなることが分かっている．よって，(構成 6) において，片方の CPU がメモリにアクセスしていると，もう一方の CPU は待機しなければならない．これに対して，(構成 7) は KVM を使用して 2 つのゲスト OS がメモリ処理を同時に行う．この際，前述した 2 段階のマッピングに加え，ホスト OS のスケジューラによって CPU のメモリアクセスの排他が行われる．このため，1 度にメモリにアクセスする CPU の個数は 1 つである．よって，Mint を用いて複数の KVM を動作し，VM を分散して動作させた場合と Vanilla linux 上で複数の VM を動作させた場合のメモリ処理の性能は同等である．

6.1.3 I/O 処理

5.4.2 項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 6.3 に示し，以下で考察する．

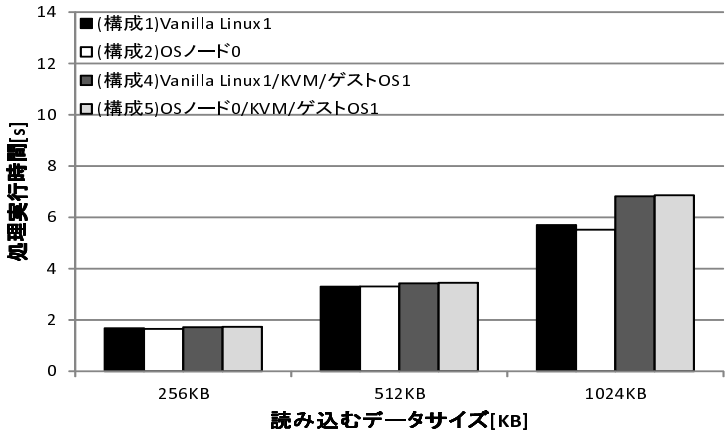
(1) 個別に処理を行った場合

(A) KVM を使用すると性能が低下する．

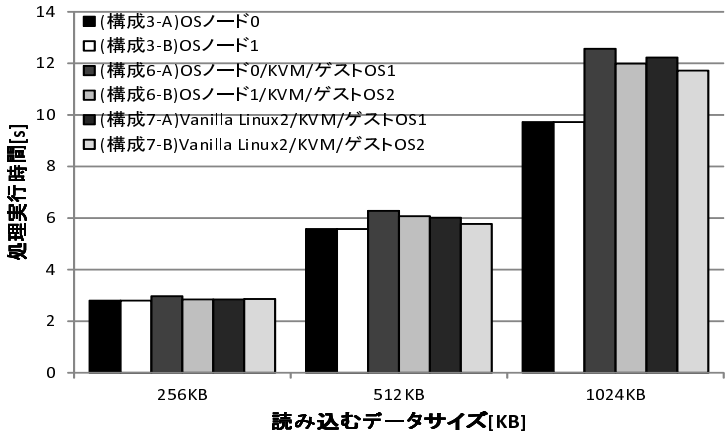
図 6.3(1) において，KVM を使用しない構成 (構成 1)，(構成 2) と比べて，それらの上に KVM を動作させる構成 (構成 4)，(構成 5) の処理実行時間が長い．具体的には，KVM を使用しない構成 (構成 1)，(構成 2) と比べて，それらの上に KVM を動作させる構成 (構成 4)，(構成 5) の処理実行時間は共に約 16 %増加している．これは，KVM では QEMU がゲスト OS の命令を物理デバイスが実行できるように I/O 命令を変更する処理が発生するためである．具体的には，ゲスト OS から I/O 命令が発行されると実 CPU が命令を解析し，KVM に例外を通知する．次に KVM は，この例外に応じて QEMU にゲスト OS から発行された命令の変換を行わせる．そして，QEMU はデバイスドライバに変換した命令を通知し，実行させる．これにより，ゲスト OS の I/O 命令の処理が完了する．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．

(B) Vanilla Linux と Mint の性能は同等である．

図 6.3(1) において，(構成 1) と (構成 2) の処理実行時間に差がない．同様に，(構成 4) と (構成 5) の処理実行時間にも差がない．このため，KVM を使用する場合



(1) 個別処理



(2) 同時処理

図　6.2　処理実行時間の比較 (メモリ)

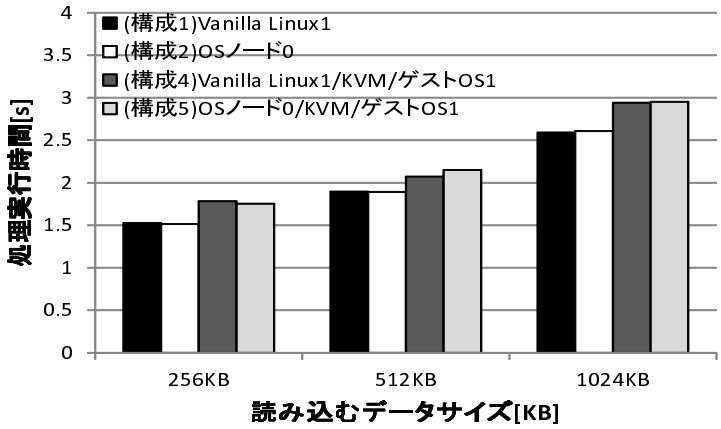
と使用しない場合の両方において Vanilla Linux と Mint では I/O 処理の性能は同等である.

(2) 同時に処理を行った場合

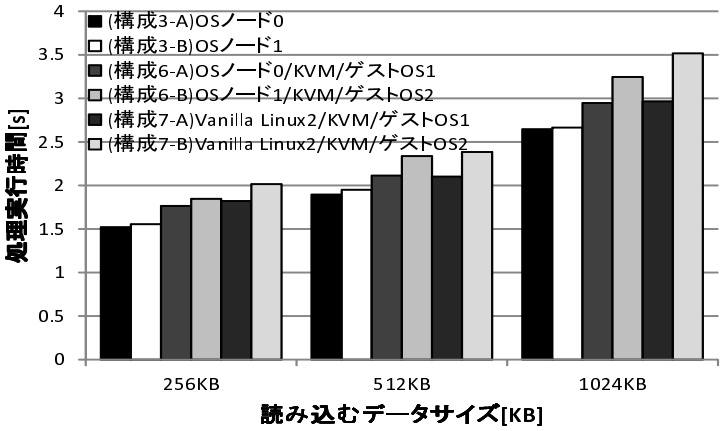
(A) ハイパーバイザ間の処理負荷の影響は小さい

図 6.3(2) における (構成 6) と (構成 7) の処理実行時間を比較すると，(構成 7) の処理実行時間が (構成 6) の処理実行時間よりも長い．具体的には，(構成 7) の処理実行時間は (構成 6) の処理実行時間に比べて約 8 %増加している．(構成 6) は OS node0 でゲスト OS1 を動作させ，OS node1 にゲスト OS2 を動作させ，同時に評価プログラムを実行している．このため，ゲスト OS の処理は OS node ごとに

1 つの CPU に割り当てられ，QEMU の処理と I/O 処理が行われる．一方，(構成 7) では Vanilla Linux 上でゲスト OS1 とゲスト OS2 を同時に動作させ，それぞれのゲスト OS で同時に評価プログラムを実行している．このため，ゲスト OS1 の処理とゲスト OS2 の処理は，ホスト OS のスケジューリングによって CPU が割り当てられて QEMU の処理と I/O 処理が行われる．これら 2 つの処理の違いは I/O 処理を行う際に CPU がデバイスドライバに処理を通知する流れである．(構成 6) は，2 つの CPU がデバイスドライバに処理を通知する際の排他制御はハードウェアが行う．一方，(構成 7) では，2 つの CPU がデバイスドライバに処理を通知する際の排他制御はホスト OS のスケジューラとハードウェアが行う．よって，Vanilla Linux の KVM 上で複数の VM が同時に I/O 処理を行う場合，ホスト OS のスケジューラが影響して処理実行時間が長くなると考えられる．したがって，異なるハイパーバイザ上の VM 間では，処理負荷の影響は小さい．



(1) 個別処理



(2) 同時処理

図　6.3　処理実行時間の比較 (I/O)

第 7 章

おわりに

本論文では，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，まず VM 方式とその問題点について述べた．次に，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，利用するハイパーバイザ，および構成について述べた．

その後，Mint で KVM を動作させ，CPU 処理，メモリ処理，および I/O 処理の基本評価を行った．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．

そして最後に，評価結果とその考察について述べた．まず，I/O 処理，CPU 処理，メモリ処理の性能は 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範囲を限定できることについて示した．なお，CPU 処理，およびメモリ処理については，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

本研究を進めるにあたり，懇切丁寧なご指導をしていただきました乃村能成准教授に心より感謝の意を表します．また，研究活動において，数々のご指導やご助言をいただいた谷口秀夫教授，山内利宏准教授，ならびに後藤佑介助教に心から感謝の意を表します．

また，日頃の研究活動において，お世話になりました研究室の皆様に感謝の意を表します．

最後に，本研究を行うにあたって，経済的，精神的に支えになった家族に感謝いたします．

参考文献

[1] Paul Barham，Boris Dragovic，Keir Fraser，Steven Hand，Tim Harris，Alex Ho，Rolf Neugebauer，Ian Pratt，and Andrew Warfield，“Xen and the Art of Virtualization,” Proc. of the 19th ACM Symposium on Operating Systems Principles, pp.164-177, 2003.

[2] Avi Kivity, Yaniv Kamay, Dor Laor, Uri Lublin and Anthony Liguori，“ kvm: the Linux Virtual Machine Monitor, ”Proceedings of the Linux Symposium, Ottawa, Ontario, pp. 225-230 (2007).

[3] Hitoshi Ueno，Satomi Hasegawa，“Virtage: Hitachi’s Virtualization Technology，” Proceeding GPC ’09 Proceedings of the 2009 Workshops at the Grid and Pervasive Computing Conference，pp.121-127 ，(2009).

[4] 株式会社日立製作所，レッドハット株式会社，“日立が業界で初めて 1 台の PC サーバ上で複数の仮想化ソフトウェアを動作させる技術を開発し，レッドハットとの連携によるサーバ仮想化ソリューションを提供開始，” http://www.hitachi.co.jp/New/cnews/month/2012/09/0913a.pdf

[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] Jeremy Sugerman，Ganesh Venkitachalam，and Beng-Hong Lim，“ 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.