修 士 論 文

題 目

Mintオペレーティングシステムにおけるデバイス移譲方式

指導教員

報 告 者

左海 裕庸

岡山大学大学院自然科学研究科電子情報システム工学専攻

平成25年2月7日 提出

要約

計算機の高性能化にともない，計算機資源を全て使用することは少なくなってきた．そこで，計算機資源を有効に活用するために，複数のOSを仮想化して1台の計算機で動作させる技術として仮想計算機方式が研究されている．しかし，この方式は，計算機の仮想化によるオーバヘッドのため，実計算機よりも性能が低下する．また，OS間での処理負荷の影響が存在する．そこで，性能の低下を抑えるため，マルチコアプロセッサ上で複数のLinuxカーネルを独立に走行させる方式としてMintオペレーティングシステムが研究開発されている．

OSの負荷は一定ではなく，時間とともに変化する．このため，負荷に応じて計算機資源を動的に割り当てることで計算機資源を効率的に使用できる．一方で，Mintでは計算機資源を各OSの起動時に静的に割り当てている．このため，計算機資源を動的に割り当てることができない．この問題を解決するために，Mintにおけるコアの動的割り当て機構が実現されている．また，MintはLinuxのLoadable Kernel Moduleを利用できる．このLKMのロード/アンロードを利用して，Mintはデバイスを移譲できる．このLKMのロード/アンロードを利用したデバイス移譲方式について，移譲時間と適用範囲について評価した．その結果，LKMによるデバイス移譲方式は，ミリ秒単位のオーダーであることが分かった．

また，LKMによるデバイス移譲方式より短い時間での移譲可能な方式について検討した．具体的には，MSI割り込みルーティング変更によるデバイス移譲方式について検討した．

次に，割り込みルーティング変更によるデバイス移譲方式の課題，対処，および実現について述べた．そして，割り込みルーティング変更によるデバイス移譲方式を評価し，LKMによるデバイス移譲方式と比較した．その結果，実現工数が大きいものの，タイムスライス間隔のような短い周期でのデバイス移譲には，割り込みルーティング変更によるデバイス移譲方式が有用であることが分かった．また，ユーザ任意のような比較的長い周期でのデバイス移譲には，LKMによるデバイス移譲方式が有用であることが分かった．

目次

図 目 次

表 目 次

第 1 章

はじめに

計算機の高性能化にともない，計算機資源を全て使用することは少なくなってきた．そこで，計算機資源を有効に活用するために，複数の OS を仮想化して 1 台の計算機で動作させる技術として VMware[1] や Xen[2] といった仮想計算機方式が研究されている．しかし，これらの方式は，計算機の仮想化によるオーバヘッドのため，実計算機よりも性能が低下する．また，OS 間での処理負荷の影響が存在する．そこで，性能の低下を抑えるため，マルチコアプロセッサ上で複数の Linux カーネルを独立に走行させる方式として Mint(Multiple Independent operating systems with New Technology)[3] オペレーティングシステムが研究開発されている．

OS の負荷は一定ではなく，時間とともに変化する．このため，負荷に応じて計算機資源を動的に割り当てることで計算機資源を効率的に使用できる．一方で，Mint では計算機資源を各 OS の起動時に静的に割り当てている．このため，計算機資源を動的に割り当てることができない．この問題を解決するために，コアの動的割り当てを可能とする方法として，Mint におけるコアの動的割り当て機構 [4] が実現されている．また，Mint は Linux カーネルをベースとして開発されているため，Linux の Loadable Kernel Module(以下，LKM) を利用できる．この LKM のロード/アンロードを利用して，Mint はデバイスを移譲できる．具体的には，1 つの Linux カーネルが当該デバイスのデバイスドライバをロードして，デバイスを利用した後，アンロードして使用を停止すれば，再び別の Linux カーネルがロード/アンロードによってデバイスを利用できる．この LKM のロード/アンロードを利用したデバイス移譲方式について，移譲にかかる時間と適用範囲について評価する．また，LKM のロード/アンロードを利用したデバイス移譲方式より短い時間での移譲可能な方式について検討

する．具体的には，割り込みルーティング変更によるデバイス移譲方式について検討する．次に，割り込みルーティング変更によるデバイス移譲方式について検討する．最後に，割り込みルーティング変更によるデバイス移譲方式を評価し，LKM のロード/アンロードを利用するデバイス移譲方式と比較する．

第 2 章

Mintオペレーティングシステム

2.1 特徴

Mint は，Linux カーネルを改変することによって開発されている．Mint は以下の特徴を持つ．

(1) 1 台の計算機上で 2 つ以上の Linux カーネルが走行する．

(2) 各 OS は，1 つ以上のコアを占有する．

(3) 全ての OS が相互に処理負荷の影響を与えない．

(4) 全ての OS が入出力性能を十分に利用できる．

これらの特徴を実現するため，1 台の計算機上でCPU，メモリ，およびデバイスといったハードウェア資源を効果的に分割し，占有する必要がある． 図 2.1 に Mint の構成例を示し，各ハードウェアの分割と占有方法を以下で説明する．

(1) CPU : マルチコアプロセッサを対象とし，コアごとに分割して 1 つ以上のコアを各 OS が占有．

(2) メモリ : 走行させる OS の数だけ実メモリを空間分割し，各 OS で占有．

(3) デバイス : デバイス単位で分割し，各 OS は指定されたデバイスのみ占有．

また，Mint おいて走行する OS は OS ノードと呼称する．

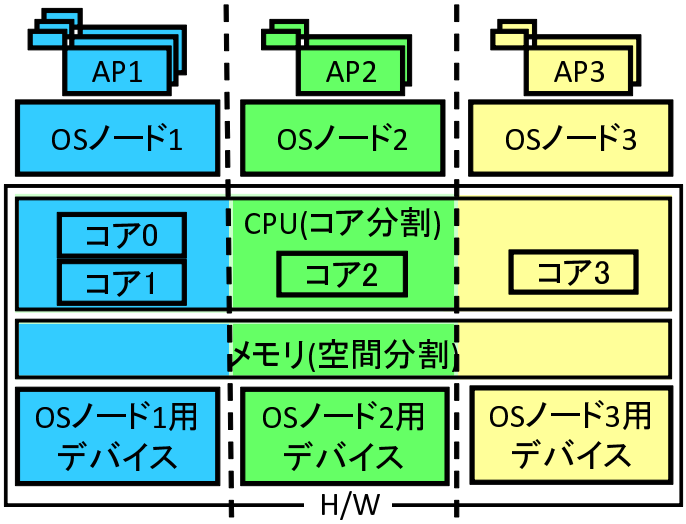

図 2.1 Mint の構成例

2.2 Mint におけるデバイス分割

図 2.2 にデバイスドライバの利用可否による入出力機器の分割例を示し，以下で説明する．PCI 等のデバイスは，デバイスドライバを利用する．このため，デバイスドライバの利用可否により，デバイスを占有制御できる．また，Linux では，デバイスに一意のデバイス番号を与えて管理しているため，この番号を使用して占有するデバイスを選択する．そこで，デバイスドライバを組み込むカーネルコードを変更し，デバイス番号によってデバイスドライバの利用の可否を決定する．つまり，占有デバイスリストにあるデバイスのみにドライバを割り当てることで占有する． 図 2.2 の例では，OS ノード 1 の占有対象のデバイスにデバイス番号 ”1000:00:00 ” が割り当てられている．このため，OS ノード 1 はデバイス番号 ”1000:00:00 ” のデバイスを使用できる．対して，OS ノード 2 はデバイス番号 ”1000:00:00 ” が占有対象のデバイスリストに含まれていないため，このデバイスを利用できない．

一方，PS/2 キーボード，VGA，およびシリアルポートのレガシーデバイスは，デバイスドライバの有無にかかわらず，認識される．このため，カーネルコードを直接変更し，認識の可否を操作することで，分割，占有する．

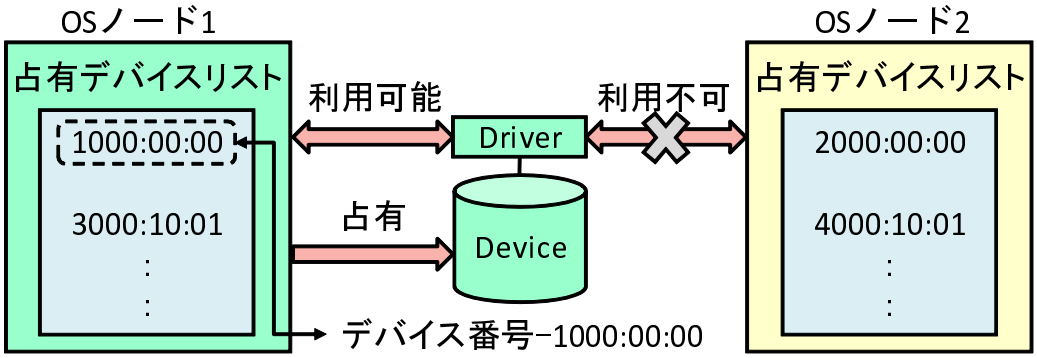

図 2.2 デバイスドライバの利用可否による入出力機器の分割

2.2.1 Mint における割り込み制御

図 2.3 に Mint における割り込み制御を示し，以下で説明する．割り込みの種類によっては，各カーネル間で割り込みを共有する必要がある．例えば，計算機内に 1 つしかなく，すべての OS に必要となるグローバルタイマからの割り込みや 1 つの割り込み線を複数のデバイスで共有する PCI デバイスからの割り込みが考えられる．これらは，デバイス分割を行っている場合でも，割り込みを共有しなければならない．図 2.3 の入出力機器の分割例では，SATA に OS ノード 1 の HDD と OS ノード 2 の HDD が接続されているため，これらは割り込みを共有しなければならない．

また，割り込みの通知先について，Linux カーネルは起動時に I/O APIC の設定を一旦クリアし，自身の占有するデバイスに対して，自身のコアに割り込みを通知するよう設定する．このとき，先に起動したカーネルが I/O APIC に設定した内容は，後に起動したカーネルに上書きされてしまう．このため，各カーネルについて，他カーネルのコアに関する設定内容を操作しないようにする．具体的には，既に I/O APIC に設定されている割り込みの割り込み先と設定しようとする割り込み先の両方に割り込みを通知するように変更する．

さらに，I/O APIC の各割り込みには，対応するベクタ番号がある．ベクタ番号は，各割り込みに対して 1 つのみ設定できるため，割り込みを共有するカーネル間で同じ値を用いる必要がある．このため，Mint の各カーネルは，既にベクタ番号が設定されている割り込みに対しては，I/O APIC からベクタ番号の設定を読み出し，同じベクタ番号を利用するように変更する．

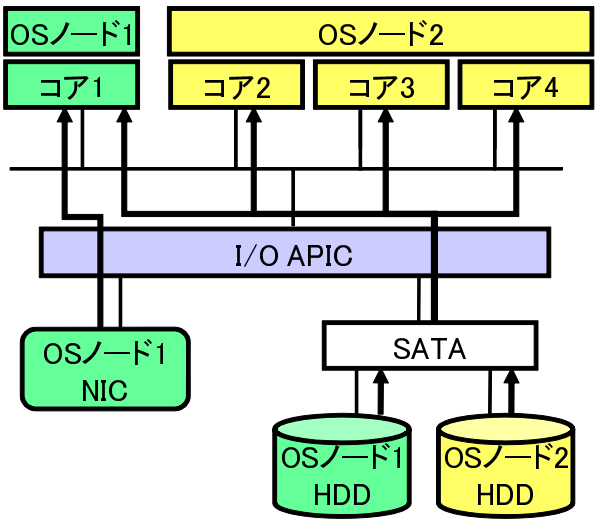

図 2.3 Mint における割り込み制御

2.3 Mint におけるデバイス移譲の目的

Mint におけるデバイス移譲とは，複数の OS ノードが走行中である状態で，ある OS ノードが占有するデバイスの占有状態を変更し，別の OS ノードで使用できるようにすることである．例えば，OS ノード 1 が NIC デバイスを占有し，走行しているとする．ここで，別の OS ノードが OS ノード 1 が占有している NIC デバイスを使用したい状況となった場合，Mint では通常，それぞれの OS ノードを再起動し，デバイスの占有状態を変更する必要がある．ここで，デバイス移譲が使用できる場合，各 OS ノードを再起動せずに NIC デバイスの占有状態を OS ノード 1 から別の OS ノードへ変更できる．

どのような契機で，どの程度の頻度で発生するのかによって，デバイス移譲の目的は異なる．デバイス移譲の契機と移譲時間に求められる性能について 2.1 に示し，以下で説明する．

(1) パケット受信ごとにデバイスを移譲する場合

例えば，NIC デバイスを移譲する場合，デバイス移譲の契機として，パケット受信時ということが考えられる．この場合，デバイス移譲の頻度はパケット受

表 2.1 デバイス移譲の契機と移譲時間

契機	デバイス移譲の時間
(1) パケット受信ごと	パケット受信間隔よりも短い間隔
(2) タイムスライスごと	タイムスライス間隔よりも短い間隔
(3) 処理負荷が高まった際の負荷分散	比較的長くてもよい
(4) ユーザの任意	比較的長くてもよい

信間隔であるため，デバイス移譲に掛かる時間は，1パケット受信するよりも短くなければならない．

(2) タイムスライスごとにデバイスを移譲する場合

例えば，デバイスを移譲する契機として，タイムスライスが考えれらる．この場合，デバイス移譲の頻度はタイムスライス間隔であるため，デバイス移譲に掛かる時間は，タイムスライス間隔よりも十分に短くなければならい．

(3) 処理負荷が高まった際の負荷分散を契機としてデバイスを移譲する場合

あるOSノードの処理負荷が高まり，負荷分散のためにデバイスを移譲することが考えられる．例えば，複数のNICデバイスを搭載している計算機にてMintを走行させる．このとき，走行する各OSノードは複数のNICデバイスを占有するとする．この状態で，あるOSノードのネットワーク処理負荷が高まった場合，別のOSノードからNICデバイスを処理負荷が高まったOSノードへ移譲するということが考えられる．この場合，デバイス移譲の頻度は(1)と(2)より少ないため，デバイス移譲に掛かる時間は，(1)と(2)と比べて，長くてもよい．

(4) ユーザの任意でデバイスを移譲する場合

例えば，ユーザがあるOSノードの占有下であるディスプレイを別のOSノード使用したいと考えた際，デバイス移譲を行うと考えられる．この場合，デバイス移譲の頻度は(1)と(2)より少ないため，デバイス移譲時間は(1)と(2)と比べて長くてもよい

次章以降では，LKMによるデバイス移譲方式と割り込みルーティング変更によるデバイス移譲方式について述べ，それぞれの適用範囲について評価する．

第 3 章

Mintにおけるデバイス移譲方式

3.1 LKMによるデバイス移譲

3.1.1 利用方法

Mintは，Linuxカーネルを改変して開発されており，Linuxの機能を利用できる．このLinuxの機能の中にデバイスドライバをLKMとする機能がある．このLKMを利用したデバイス移譲方法について図3.1に示し，以下で説明する．

(1) 1つのOSノードが当該デバイスのデバイスドライバをロードしてデバイスを利用する．

(2) デバイスを利用したOSノードが当該デバイスのデバイスドライバをアンロードし，デバイスの利用を停止する．

(3) 別のOSノードが当該デバイスのデバイスドライバをロードし，デバイスを利用する．

3.1.2 評価

LKMのロード/アンロードを利用したデバイス移譲方式について，NICデバイスドライバを対象にロード/アンロードの時間を測定した．LKMのロード処理の流れについて， 図3.2に示し，以下で説明する．

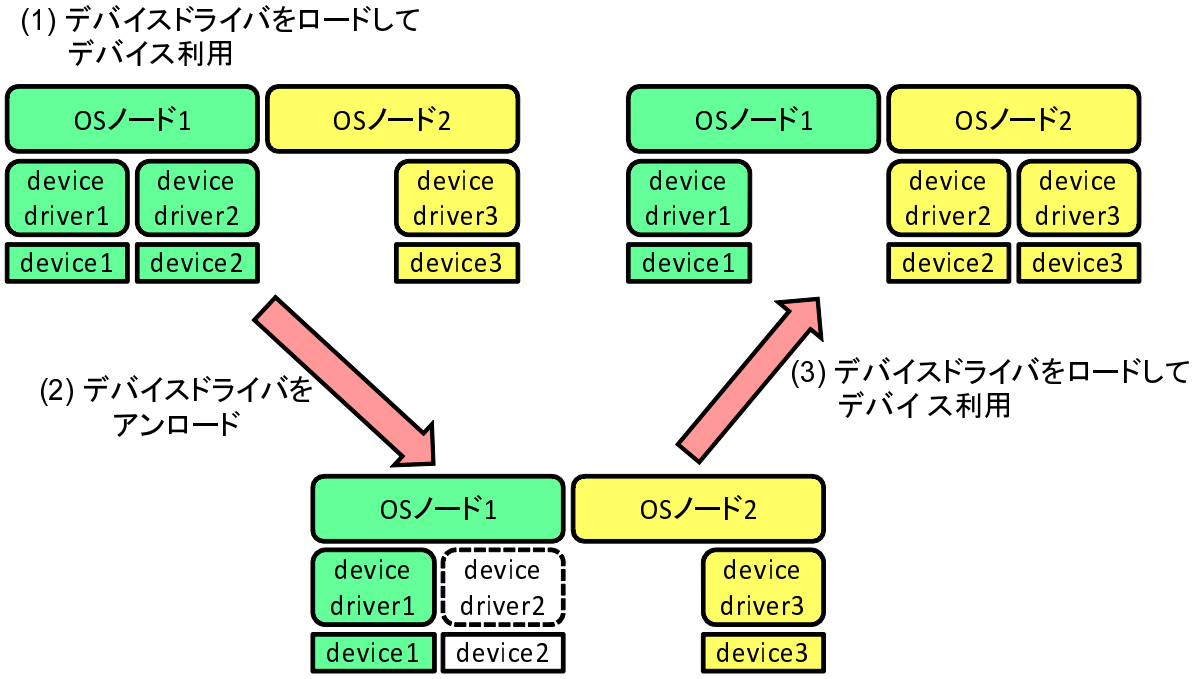

図 3.1 LKM によるデバイス移譲

(1) modprobe コマンドを実行する.

(2) init_module システムコールの呼び出し前に前処理を行う.

(3) init_module システムコールを呼びだす.

(4) init_module システムコールによって，LKM のロード処理とデバイスの初期設定を行う.

(5) init_module システムコール呼び出し終了後の後処理を行う.

(6) modprobe コマンドの実行が終了する.

LKM のアンロード処理の流れは，LKM のロード処理と同様であり，init_module システムコールの代わりに delete_module システムコールが呼ばれる．delete_module システムコールは，デバイスの終了処理と LKM のアンロード処理を行う.

図 3.2 に示した各処理の処理時間を表 3.1 に示す．また，表??にこの方式の利点と欠点を示す.

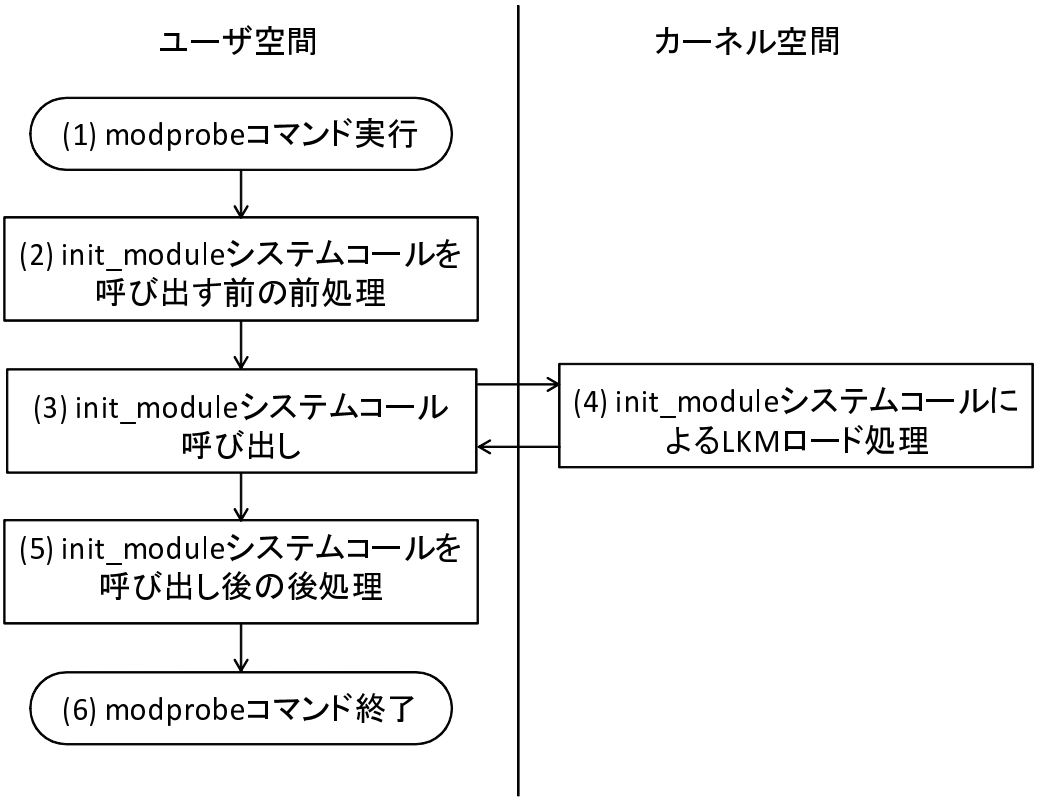

図 3.2 LKM のロード処理の流れ

表3.1より，この方式でのデバイス移譲時間はミリ秒単位のオーダーである．また，Linuxカーネルのタイムスライス間隔は，ミリ秒単位のオーダーである．これらのことから，2.3節で示したタイムスライス契機でのデバイス移譲は行えない．また，タイムスライス間隔はパケット受信間隔よりも長い周期であると考えられるため，パケット受信間隔でのデバイス移譲も行えない．逆に，負荷分散を契機としてのデバイス移譲とユーザ任意でのデバイス移譲には対応できると考えられる．これは，負荷分散を契機としてのデバイス移譲とユーザ任意でのデバイス移譲は，パケット受信間隔でのデバイス移譲やタイムスライス間隔でのデバイス移譲よりも比較的長い周期でもよく，ミリ秒単位のオーダーであれば十分であると考えるからである．

表 3.1 NICデバイスドライバのロード，アンロード時間

通番	測定箇所	時間 (ms)	合計 (ms)
3.1-1	(2)modprobe コマンド (ロード時の前処理)	0.60	
3.1-2	(4)init_module システムコール	77.8	78.4
3.1-3	(5)modprobe コマンド (ロード時の後処理)	0.0076	
3.1-4	(2)modprobe コマンド (アンロード時の前処理)	0.54	
3.1-5	(4)delete_module システムコール	254.3	254.8
3.1-6	(5)modprobe コマンド (アンロード時の後処理)	0.004	

3.2 LKMによらないデバイス移譲

3.2.1 デバイスからの割り込みの流れ

割り込み発生時におけるデバイスからCPUまでの割り込みの流れを 図3.3 に示し，以下で説明する．

(1) device A が割り込みを発行する．

(2) I/O APICは，割り込みが発生したピンのピン番号mに対応するリダイレクションテーブルのエントリを参照し，割り込みベクタ番号nへの変換と割り込み通知先の決定を行う．ここで，リダイレクションテーブルとは，I/O APICの各ピンごとに存在し，割り込みベクタ番号や割り込み通知先を格納するテーブルである．今回の例では，割り込みベクタ番号n，割り込み通知先はコア0となる．

(3) ベクタ番号nを割り込み通知先であるコア0のLocal APICに通知する．

この割り込みの流れから分かるように，指定したコアへ割り込みを通知することができる．Mintでは，コアを分割占有するため，割り込み通知先を変更することで特定のOSノードにのみ割り込みを通知できる．このことから，割り込み通知先を特定のOSノードのみに制限することで，デバイスを占有できると考えられる．よって，割り込み通知先変更により，デバイスの占有状態を変更できると考えられる．

3.2.2 割り込みルーティング変更によるデバイス移譲

前項より，前節で述べたデバイス移譲方式とは異なる方式として，以下の方式が考えられる．

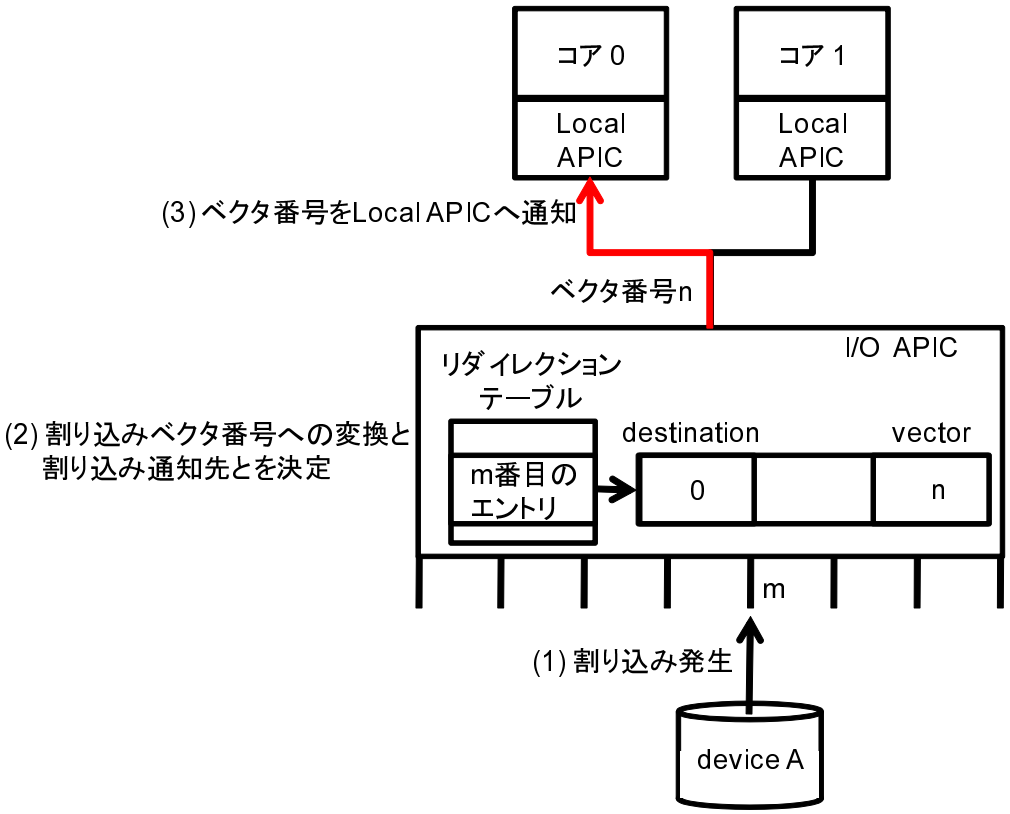

図 3.3 割り込みの流れ

(方式) 各 OS ノードにデバイスドライバをロードしておき，割り込みルーティングの変更によって占有状態を変更する方式.

各 OS ノードに移譲対象デバイスのデバイスドライバをあらかじめロードしておく．Mint では，割り込みは指定されている OS ノードのみに通知されるため，デバイスを占有する OS ノードのみに割り込みを通知するように割り込みルーティングを変更する．デバイスを移譲する際には，デバイスからの割り込み通知先を変更することで，デバイスの占有状態を変更する．

しかし，この方法には次の問題点が存在する．

(問題点) pin-base(割り込み線を使用する) 割り込みにおいて，割り込み線を共有するデバイスが存在する.

割り込み線を共有する複数のデバイスが存在し，それぞれのデバイスが異なる OS ノードの占有下である場合，割り込み通知先の制限ができない．また，デバ

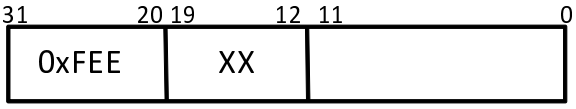

図 3.4 Message Address Register

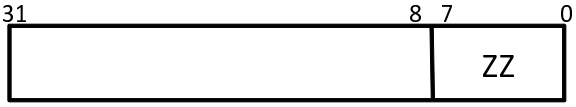

図 3.5 Message Data Register

イスからの割り込み通知先を変更すると，割り込み線を共有するデバイスを意図せず移譲してしまう可能性がある．

この問題点に対処するために，Message Signaled Interrupt(以下，MSI) を利用する方法が考えられる．MSIは，割り込み線を使用せず，特定のデータを特定のアドレスへ書き込むことによって発行される割り込みであり，PCI2.2以降の規格で使用できる．MSIは，2種類の割り込み発行に関わるレジスタがPCI Deviceごとに存在する．この2つのレジスタ，Message Address RegisterとMessage Data Registerのレイアウトをそれぞれ図3.4と図3.5に示し，以下で説明する．

(1) Message Address Register

Message Address Registerは，PCI Deviceごとに存在し，Message Data Registerに格納するデータを書き込むアドレスを指定するレジスタである．このレジスタの12〜19ビットは，割り込み通知先情報(XX)を表す.

(2) Message Data Register

Message Data Registerは，PCI Deviceごとに存在し，Message Address Registerで指定するアドレスへ書き込むデータを格納するレジスタである．このレジスタの0〜7ビットは，割り込みベクタ番号(ZZ)を表す.

これら2つのレジスタを利用して，PCI DeviceはMSIを発行する．MSIによる割り込みの流れについて， 図3.6に示し，以下で説明する．

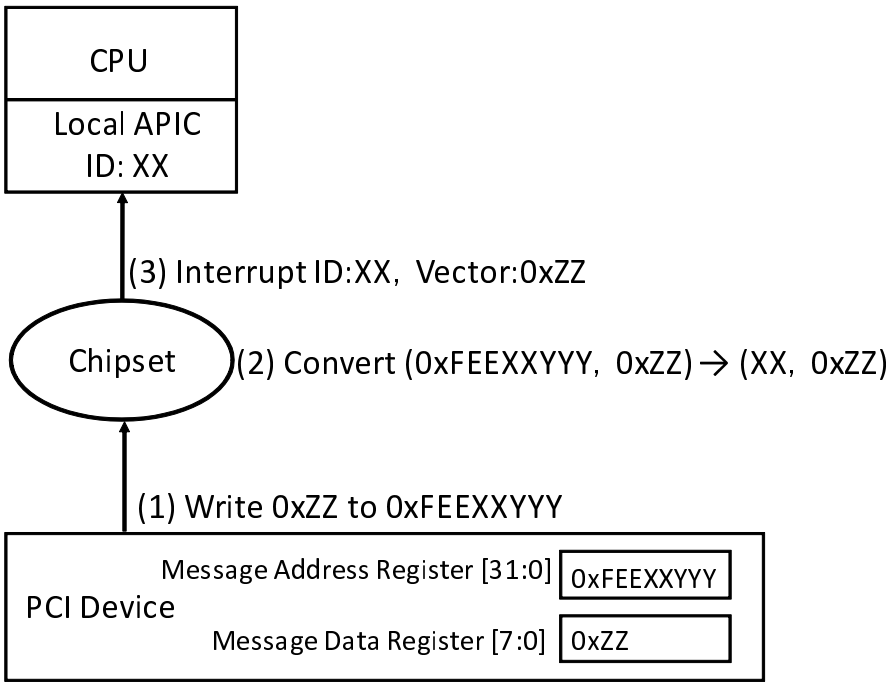

図 3.6 MSI による割り込みの流れ

(1) PCI Device は，Message Data Register に格納する割り込みベクタ番号 (0xZZ) を含むデータを Message Address Register に格納する割り込み通知先情報 (XX) を含むアドレス (0xFEEXXYYY) へ書き込むことにより，Chipset へ割り込み発生を通知する．

(2) Chipset は，PCI Device が書き込んだデータとアドレスをベクタ番号が 0xZZ であり，割り込み通知先の ID が XX である CPU への割り込みへ変換する．

(3) Chipset は，ベクタ番号が 0xZZ である割り込みを Local APIC の ID が XX である CPU へ通知する．

MSI による割り込みは，PCI Device ごとに存在する割り込み通知先の情報を含むレジスタをデバイスドライバが設定する．このため，この部分をユーザがカーネルに依頼して変更することにより，割り込み通知先を変更できる．

3.3 MSIによるデバイス移譲方式の実現

3.3.1 課題

MSI によるデバイス移譲を実現するためには，以下の課題がある．

(課題 1) 移譲するデバイスの指定方法

移譲するデバイスを指定する方法が必要である．

(課題 2) 割り込み通知先の指定と書き換え方法

割り込み通知先として移譲先の OS ノードを指定する方法と割り込み通知先の書き換え方法が必要である．

(課題 3) 割り込みベクタ番号の変更

各 OS ノードごとに割り込みベクタ番号を設定するため，デバイス移譲時に各 OS ごとに割り込みベクタ番号を変更する必要がある．

(課題 4) デバイス固有の処理

デバイスによって，割り込み通知先変更以外の処理が必要となる．

3.3.2 対処

3.3.1 項で示した (課題 1) から (課題 4) への対処として，(対処 1) から (対処 4) を示す．

(対処 1) デバイス ID による移譲デバイスの指定

Linux は，デバイスごとに一意のデバイス ID を割り当て，管理している．このため，移譲するデバイスの指定にはデバイス ID を用いる．また，デバイス ID を引数として，デバイス ID に対応する pci_dev 構造体を取得できる．この pci_dev 構造体には，IRQ 番号やデバイスのレジスタへアクセスするためのアドレスといったものが格納されている．

(対処 2) 移譲時における割り込み通知先の指定と書き換え

Mint では，各 OS ノードが占有するコアに一意の Logical APIC ID を割り振る．ここで，Logical APIC ID とは，割り込み通知先を指定するために用いられる値である．MSI の割り込み通知先もこの値で指定できる．このため，移譲先の OS

ノードが占有するコアに割り振られたLogical APIC IDを指定し，割り込み通知先を指定する．また，割り込み通知先の書き換え方法としては，PCI Deviceのレジスタの読み書きを行うLinuxの既存機構を用いる．具体的には，pci_dev構造体よりPCI Deviceのレジスタ読み書きに使用するベースアドレスを取得する．そして，PCI Deviceのレジスタ読み書きを行う関数を用いて，Message Address Registerへ割り込み通知先を設定する．

(対処 3) 移譲時における割り込みベクタ番号の変更

各OSノードごとに割り込みベクタ番号を設定するため，デバイス移譲時に割り込みベクタ番号を移譲先OSノードが設定する割り込みベクタ番号に変更する．具体的には，pci_dev構造体よりIRQ番号を取得し，これを割り込みベクタ番号に変換する．(対処2) と同様に，移譲するPCI DeviceのMessage Data Registerへ設定する．

(対処 4) デバイスドライバによる初期設定の解析と変更

デバイスによって，各OSノードごとに設定する値が存在する場合がある．これについて，デバイスドライバの初期設定を解析し，各OSノードごとに設定する値が存在するか否かを把握する．各OSノードごとに設定する値が存在する場合，デバイス移譲時に移譲先OSノードが設定する値へ変更する．

3.4 NICデバイス移譲の実現

3.4.1 NICデバイス移譲時に必要な固有処理

MSIによるデバイス移譲方式をNICデバイスへ実装した．3.3.2項にて述べたように，デバイスによって，各OSノードごとに設定する値が存在する場合がある．NICデバイスはこの場合に当てはまり，各OSノードごとに設定しなければならない値が存在する．今回使用したNICデバイスとNICデバイスドライバについて，表3.2に示す．また，各OSノードごとに設定しなければならない値について，以下で説明する．

(1) Transmit Normal Priority Descriptors

送信バッファの管理に利用するTx Descriptor Ringのスタートアドレスを設定するレジスタである．

表 3.2 使用したNICデバイスとデバイスドライバ

使用したNICデバイス	RTL8111/8168B PCI Express Gigabit Ethernet controller
使用したデバイスドライバ名	r8169

(2) Receive Descriptor Start Address

受信バッファの管理に利用するRx Descriptor Ringのスタートアドレスを設定するレジスタである．

NICデバイスの送受信バッファとこれを管理するDescriptor Ringは，各OSノードごとに確保するため，デバイス移譲時に移譲先OSノードのものに変更する．具体的には，pci_dev構造体よりrtl8169_private構造体を取得する．この構造体は，今回使用したデバイスドライバ固有の構造体であり，Transmit Normal Priority DescriptorsとReceive Descriptor Start Addressへのオフセットや設定した値といったものを格納している．この構造体を用いて，Transmit Normal Priority DescriptorsとReceive Descriptor Start Addressの値を移譲先OSノードのものに変更する．

3.4.2 送受信処理に追加する処理

NICデバイスを移譲する場合，3.4.1項で述べた移譲時に必要な処理以外に送受信処理への変更も必要である．まず，図3.7に受信処理の流れを示し，以下で説明する．

(1) NICデバイスがパケットを受信する．

(2) NICデバイスはRx Descriptorに情報を書きこむ．このとき，NICデバイスは受信した総パケット数をカウントしており，このカウントに応じてRx Descriptor Ringの先頭からずらした位置に情報を書きこむ．

(3) NICデバイスは受信完了割り込みを発行する．

(4) NICデバイスドライバが受信処理のために，Rx Descriptorの情報を読みこむ．このとき，NICデバイスドライバは受信した総パケット数をカウントしており，このカウントに応じてRx Descriptor Ringの先頭からずらした位置の情報を読みこむ．

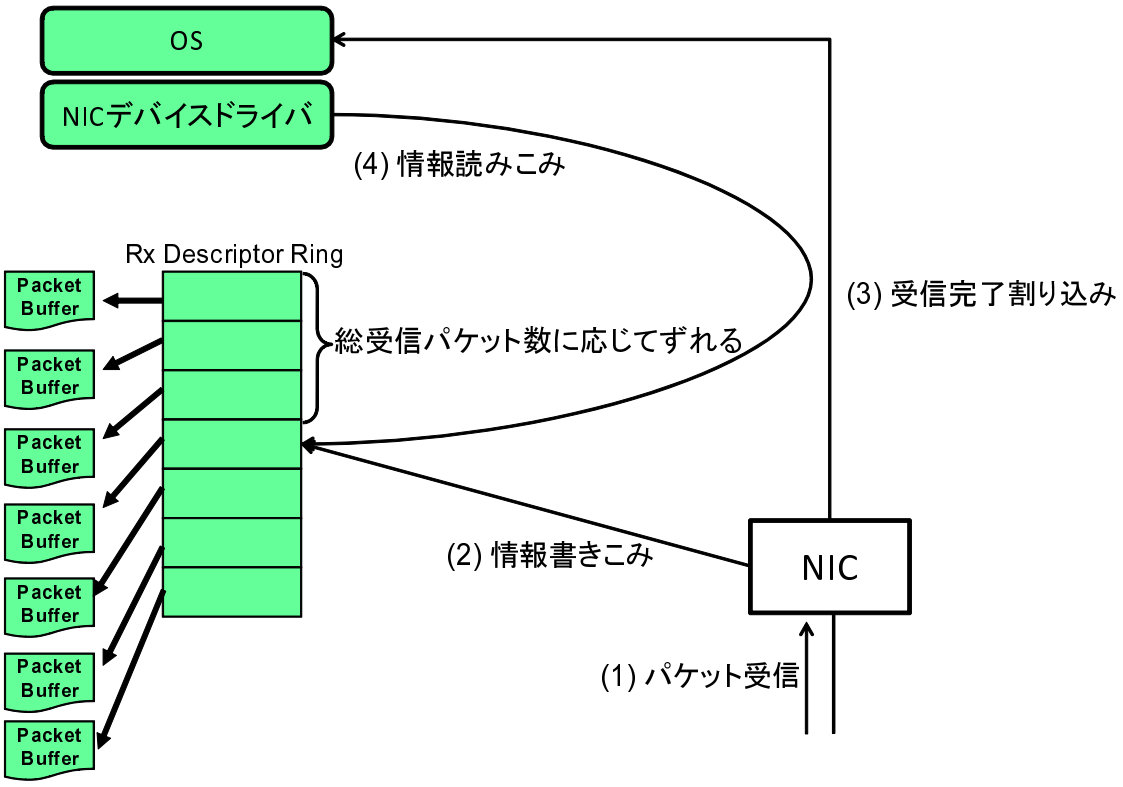

図 3.7 受信処理の流れ

MSI によるデバイス移譲方式を用いた際に受信処理で起こる問題について図 3.8 に示し，以下で説明する．

(1) NIC デバイスはパケットを受信した際，NIC を占有している OS ノードの Rx Descriptor へ情報を書きこむ．このとき，NIC デバイスは受信した総パケット数をカウントしており，このカウントに応じて Rx Descriptor Ring の先頭からずらした位置に情報を書きこむ．

(2) NIC デバイスを占有している OS ノードの NIC デバイスドライバは，受信処理のために Rx Descriptor の情報を読みこむ．このとき，自身が NIC デバイスを占有している状態で受信した総パケット数に応じて，Rx Descriptor Ring の先頭からずらした位置の情報を読みこむ．

(3) (2) において，NIC デバイスが移譲後の状態である場合，NIC デバイスドライバがカウントする総受信パケット数と NIC デバイスがカウントする総受信パケット

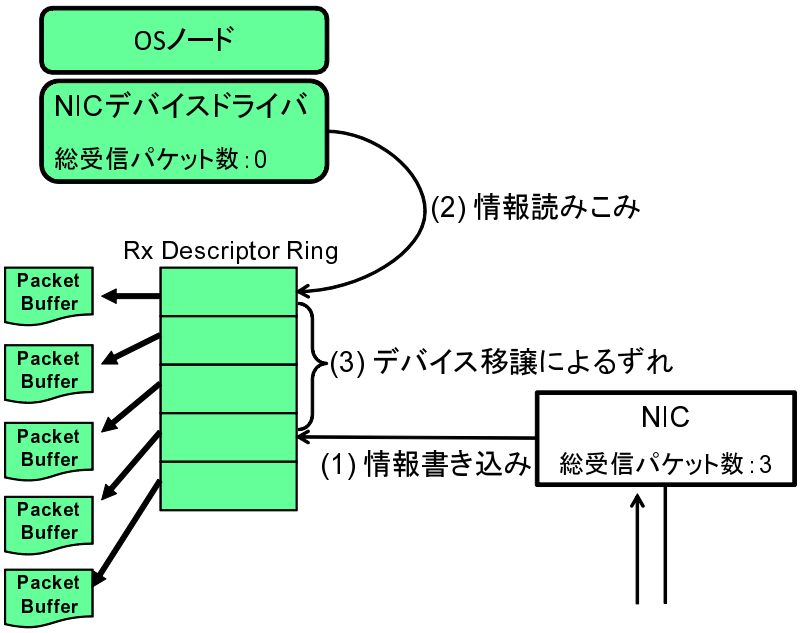

図 3.8 移譲後の NIC デバイスにおける問題

数にずれが発生する可能性がある．この結果，NIC デバイスが情報を書きこんだ Rx Descriptor と NIC デバイスドライバが受信処理で情報を取得する Rx Descriptor が一致しないことがある．このため，受信処理に失敗する可能性がある．

送信処理でも同様に，NIC デバイスがカウントする総送信パケット数と NIC デバイスドライバがカウントする総送信パケット数にずれが発生する可能性がある．このため，送信処理でも NIC デバイスと NIC デバイスドライバ間で情報をやり取りする Tx Descriptor が一致しない可能性がある．この問題に対処するために，NIC デバイス移譲後の最初の送受信処理では，NIC デバイスと OS ノードの送受信パケット数に関するずれを補正した．具体的には，NIC デバイス移譲後の最初の総受信処理では，すべての Rx Descriptor と Tx Descriptor をチェックし，NIC デバイスが情報を書きこんだ Descriptor を特定する．そして，この Descriptor へのオフセットを記録し，以降の送受信処理にて使用する．

3.5 MSIによるデバイス移譲方式で期待される効果

MSI によるデバイス移譲方式は，主としてデバイスのレジスタ読み書きでデバイス移譲を行う．このため，LKM のロード/アンロードによるデバイス移譲方式のように，デバイスの初期化/終了処理を実行しない．よって，MSI によるデバイス移譲方式によって期待される効果として，デバイス移譲にかかるオーバヘッドが LKM のロード/アンロードによるデバイス移譲に比べて少なく，LKM によるデバイス移譲方式では対応できなかった，パケット受信間隔でのデバイス移譲やタイムスライス間隔でのデバイス移譲に対応できるのではないかと考える．

第 4 章

評価

4.1 評価環境

本論文で述べたデバイス移譲方式について，デバイス移譲時間について評価する．評価環境を表 4.1 に示す.

4.2 デバイス移譲時の測定箇所と結果

MSI によって割り込みルーティング変更を変更し NIC デバイスを移譲する流れについて， 図 4.1 に示し，以下で説明する．

(1) デバイス移譲のためのシステムコールを発行する．

(2) デバイス ID を用いて，NIC デバイスに対応するデータを取得する．

表 4.1 評価環境

OS	fedora 14
Linux Kernel	3.0.8
CPU	Intel Core i7(2.8GHz)
使用した NIC デバイス	RTL8111/8168B PCI Express Gigabit Ethernet controller
使用したデバイスドライバ名	r8169

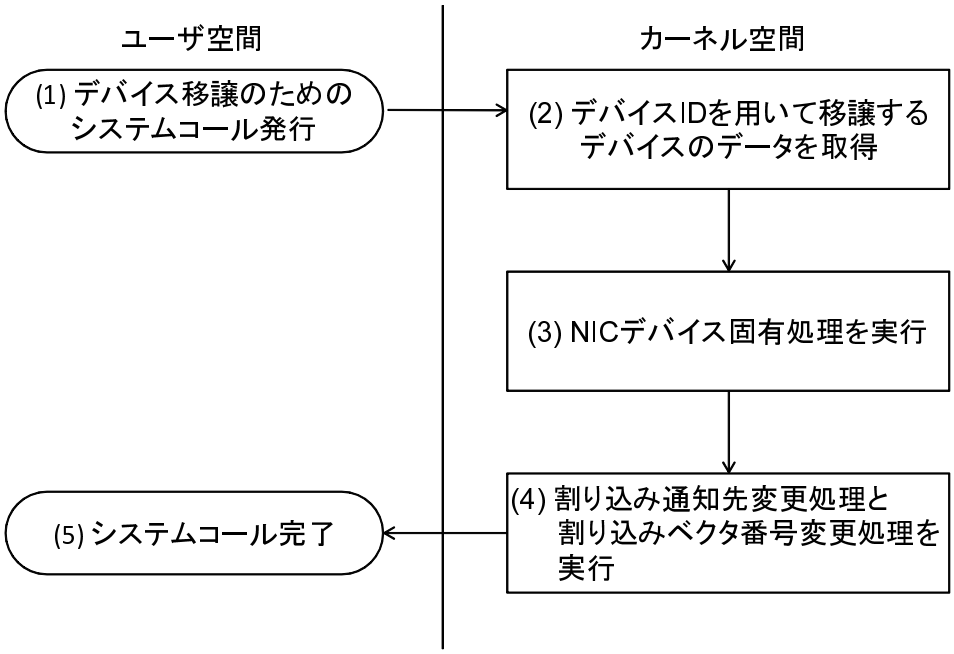

図 4.1: MSIによって割り込みルーティング変更を変更しNICデバイスを移譲する流れ

(3) NICデバイス固有の処理として，Tx Descriptor RingとRx Descriptor Ringのスタートアドレスを移譲後のOSノードのものに変更する．

(4) MSIの割り込み通知先と割り込みベクタ番号を移譲後のOSノードのものに変更する．

(5) デバイス移譲のためのシステムコールが完了する．

図4.1に示した各処理の処理時間を表4.2に示す.

考察

表4.2より，MSIによるデバイス移譲は，マイクロ秒単位のオーダーで行える．Linuxのタイムスライスはミリ秒単位のオーダーであり，MSIによるデバイス移譲はタイムスライス間隔より十分小さい．このため，タイムスライス間隔でのデバイス移譲に対応できる．また，タイムスライス間隔でのデバイス移譲に対応できるため，負荷分散によるデバイス移譲やユーザの任意によるデバイス移譲にも対応できる．しかし，MSIによるデバイス移譲方式は，デバイスドライバの解析が必要となるため，工数が掛かる．このため，MSIによるデバイス移

表 4.2 MSIによるデバイス移譲方式のデバイス移譲時間

通番	測定箇所	時間 (μ s)	合計 (μ s)
4.2-1	(2) デバイスIDを用いてNICデバイスのデータの取得	5.26	
4.2-2	(3) NICデバイス固有処理	1.68	9.08
4.2-3	(4) 割り込み通知先変更処理と割り込みベクタ番号変更処理	3.65	

表 4.3: LKMによるデバイス移譲方式とMSIによるデバイス移譲方式の対応可能な契機

	LKMによるデバイス移譲方式	MSIによるデバイス移譲方式
対応可能な契機	処理負荷が高まった際の負荷分散 ユーザの任意	パケット受信ごと タイムスライスごと 処理負荷が高まった際の負荷分散 ユーザの任意

譲方式の対象とするデバイスは，本当にこの方式が必要であるか否かを十分に検討しなければならない．

4.3 LKMによるデバイス移譲方式とMSIによるデバイス移譲方式の比較

LKMのロード/アンロードによるデバイス移譲方式とMSIによるデバイス移譲方式の対応可能な契機を表4.3に示す．

表4.3より，MSIによるデバイス移譲方式は，LKMによるデバイス移譲方式と比べて，対応可能な契機は多い．しかし，Linuxカーネルへの改変とデバイスドライバごとに解析が必要であるため，工数が掛かることを考えるとタイムスライス間隔でのデバイス移譲が必要にならないデバイス移譲方法には，向いていない．

対して，LKMによるデバイス移譲方式は，MSIによるデバイス移譲方式と比べて，対応可能な契機は少ない．しかし，Linuxカーネルへの改変が不要であるため，工数

表 4.4: LKMによるデバイス移譲方式とMSIによるデバイス移譲方式の適用可能な契機

	LKMによるデバイス移譲方式	MSIによるデバイス移譲方式
適用可能な契機	処理負荷が高まった際の負荷分散 ユーザの任意	パケット受信ごと タイムスライスごと

が掛からない.

これらのことを考えると，表4.4に示すように各方式の適用可能な契機が決まる.

第 5 章

関連研究

関連研究として，擬似NICを用いた通信監視[5]がある．擬似NICは，1台の計算機上で2つのOSが走行するTwinOSにおいて，一方のOSが占有するNICデバイスを仮想化したインタフェースを用いて，他方のOSへ提供する機能である．一方のOSが占有するNICデバイスを他方のOSへ提供する点において，本研究と類似している．しかし，本研究とは異なり，NICデバイスの占有状態を変更せず，仮想化インタフェースを通じて他方のOSへ提供する点が異なる．

通信デバイスドライバを動的に入れ替える手法として，通信デバイスドライバの動的更新手法[6]がある．これは，通信デバイスドライバのロード/アンロードを行わずに，動作中の通信デバイスドライバを更新する手法である．新旧の通信デバイスドライバの実行プログラムを一時的に共存させたうえで，旧ドライバから新ドライバに制御を切り替えるタイミングで，旧ドライバの持つ動作状態も新ドライバへ引き継がせている．通信デバイスドライバをロード/アンロードせずに状態を変更するという点において，本研究と類似している．しかし，デバイスドライバそのもののプログラムを動的に変更するという点が本研究とは異なる．

1台の計算機上で複数のOSを走行させる研究として，Twin-Linux[7]がある．これは，マルチコアの異なるコア上で異なるOSの複製を同時に走行させる機構である．多くのOSは，サーバとしての機能，相互サービス，リアルタイム処理のすべてを同時に提供することはできない．しかし，Twin-Linuxでは，各OSに役割に応じたサービスを走行させることが可能である．

SHIMOS[8]は，プロセッサやメモリといった資源を分割し，OSにそれぞれ占有させることにより，同時に複数のOSを動作させる機構である．SHIMOSでは，マルチコ

アプロセッサを分割し，分割したコアを各 OS に割り当てることで複数の OS を走行できる．実メモリの分割方法については，各 OS に割り当てるメモリマップの方式が Mint と異なっている．入出力機器の分割方法については Mint と同様に静的に決定している．

第 6 章

おわりに

Mint におけるデバイス移譲方式に関して，LKM のロード/アンロードを用いたデバイス移譲方式の移譲時間を評価した．また，LKM によるデバイス移譲方式より短い時間での移譲可能な方式について検討した．具体的には，割り込みルーティング変更によるデバイス移譲方式を検討した．この際，割り込み線を共有するデバイスが存在する場合，デバイス移譲時に意図しないデバイスの移譲が発生する可能性がある．この問題に MSI を利用することで対処した．また，デバイス移譲の実装例として，NIC デバイスの移譲を実現した．そして，割り込みルーティング変更によるデバイス移譲方式によるデバイス移譲時間を測定した．測定結果より，LKM のロード/アンロードを利用したデバイス移譲方式と割り込みルーティング変更によるデバイス移譲方式とを比較し，それぞれの適用可能な契機について評価した．結果，タイムスライス間隔のような短い周期でのデバイス移譲には，実現工数が大きいものの，割り込みルーティング変更によるデバイス移譲方式が有用であることが分かった．また，ユーザ任意のような比較的長い周期でのデバイス移譲には，LKM によるデバイス移譲方式が有用であることが分かった．

謝辞

本研究を進めるにあたり，懇切丁寧なご指導をして頂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，数々のご指導やご助言を頂きました谷口秀夫教授，山内利宏准教授，および後藤佑介助教に厚く御礼申し上げます．最後に，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ならびに本研究を行うにあたり，経済的，精神的な支えとなった家族に感謝いたします．

参考文献

[1] Jeremy Sugerman，Ganesh Venkitachalam，and Beng-Hong Lim，“Virtualizing I/O Devices on VMware Workstation’s Hosted Virtual Machine Monitor，” Proc. of the General Track: 2002 USENIX Annual Technical Conference，pp.1-14，2001.

[2] P.Barham，B.Dragovic，K.Fraser，S.Hand，T.Harris，A.Ho，R.Neugebauer，I.Pratt，and A. Warfield. Xen and the art of virtualization. In Proceedings of the ACM symposium on Operating Systems Principles，pages 164-177，October 2003.

[3] 千崎 良太，中原 大貴，牛尾 裕，片岡 哲也，粟田 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).

[4] 池田 騰，乃村 能成，谷口 秀夫，“Mintオペレーティングシステムにおけるコア管理機能の実現，”情報処理学会研究報告，vol.2012-OS-122，no.3 (2012.7).

[5] 山本 裕馬，乃村 能成，谷口 秀夫，“TwinOSにおける通信の監視方式と基本性能評価，”情報処理学会研究報告，vol.2006, no.44, pp.53-60 (2006.5).

[6] 田中 裕之，乃村 能成，谷口 秀夫，“通信デバイスドライバの動的な更新手法の提案，”情報処理学会研究報告，vol.2007，no.27，pp.13-18 (2007.3)

[7] Adhiraj Joshi，Swapnil Pimpale，Mandar Naik，Swapnil Rathi，Kiran Pawar，“Twin-Linux:Running different Kernels on separate cores of a multicore system，” Linux Symposium，pp.101-108，2010.

[8] T. Shimosawa，H. Matsuba，Y. Ishikawa，“Logical Partitioning without Architectural Supports，” Proc. of the 2008 Annual IEEE International Computer Software and Applications Conferece, pp.355-364, 2008.

発表論文

[1] 左海 裕庸，乃村 能成，谷口 秀夫，“ Mint オペレーティングシステムにおけるデバイス移譲方式，”電子情報通信学会 2012 年総合大会 情報・システム講演論文集 1, p.84 (2012.3).