情報処理学会研究報告
IPSJ SIG Technical Report

Vol.2010-SLDM-144 No.31
Vol.2010-EMB-16 No.31
Vol.2010-MBL-53 No.31
Vol.2010-UBI-25 No.31
2010/3/27

リアルタイム性保証のためのI/O割込み管理手法

永島 力[†1] 渡邉和樹[†2] 茂田井寛隆[†3]
片山吉章[†3] 毛利公一[†2]

近年，組込みシステム内に複数の計算機が搭載され，製品サイズの拡大や配線の複雑化，製品コストの増加といった問題が起きるようになった．この背景から，我々は，システム内の複数の計算機を統合し，RT-OS と高機能 OS を動作可能とするリアルタイム仮想計算機モニタ「Natsume」の開発を行っている．本論文では，既存仮想計算機モニタにおける問題点を述べ，Xen の I/O デバイスについて PCI Pass-through を用いた性能評価と割込み管理手法について報告し，I/O 割込み管理手法を提案する．

A method of managing I/O interrupt for guarantee Real-timeness

Chikara Nagashima,[†1] Kazuki Watanabe,[†2]
Hirotaka Motai,[†3] Yoshiaki Katayama[†3]
and Koichi Mouri[†2]

Recently, multiple units of computer system are mounted in a embedded system. It causes many problems. For example, its size may be bigger. Its structure of circuit may be more complicated. To solve these problems, we have been developing a real-time virtual machine monitor called Natsume that real-time OS and high functional OS can run on it. In this paper, we discuss problems of existing virtual machine monitors, evaluation of PCI Pass-through of Xen, and a method of managing I/O interrupt for realtimeness.

†1 立命館大学大学院 理工学研究科
Graduate School of Science and Engineering, Ritsumeikan University
†2 立命館大学 情報理工学部
College of Information Science and Engineering, Ritsumeikan University
†3 三菱電機株式会社 情報技術総合研究所
Information Technology R&D Center, Mitsubishi Electric Corporation

1. はじめに

近年，情報家電製品や携帯端末，各種制御装置といった組込みシステムの分野では，従来のように単にリアルタイム性や信頼性のみを提供する製品ではなく，ユーザインタフェースやアプリケーションなどの高機能な処理をあわせて提供する製品のニーズが拡大しており，そのソフトウェア開発も大規模化している．一方，短い製品ライフサイクルへの対応や，開発期間の短期化といったニーズから，全ての組込みソフトウェアにおいて生産性の向上が求められている[1)]．さらに，μITRON や VxWorks などに代表される，従来のリアルタイム OS(以下，RT-OS) では特定の機能だけを持ち，その構造を単純化することでリアルタイム性を実現してきた[2)]．そのため，ひとつの計算機内で，リアルタイム性や信頼性を維持しながら，高機能さも持った製品を実現することはリソース管理の観点で困難であり，開発期間やコストの増大につながっている．

このような問題に対して，組込みシステムの分野では，一つの製品内に処理内容ごとに分けた複数の計算機を搭載する手法が採られている．これによってそれぞれの OS(または RT-OS) 内の処理が単純化され，リアルタイム性や信頼性を従来どおり確保している．しかし，計算機を複数搭載することによる製品サイズの拡大や，計算機間の配線の複雑化，製品単価の増加といった問題を抱えており，組込み分野における仮想計算機モニタ (以下，VMM) の適用が考えられている．

以上の背景から，代表的な VMM のひとつである Xen[3)] をベースに，高機能な処理を行う OS とリアルタイム性を保証する RT-OS が共存動作することを可能にするリアルタイム仮想計算機モニタ「Natsume」の開発を行っている．しかし，Xen を含めた既存の VMM は本来リアルタイム処理向けに開発されていないため，RT-OS がリアルタイム性を維持するうえで，計算機資源の仮想化による処理のオーバヘッド軽減，ゆらぎの抑制が課題である．この課題に対して，Natsume では，RT-OS に対して割当て資源を固定化し，計算機資源の利用にかかるオーバヘッドを抑えたリアルタイム仮想計算機 (以下，RT-VM) を用意する．

本論文では，Natsume を実現するにあたって，CPU 割当ての課題や I/O 処理の問題，メモリ割当ての問題などいくつかの問題に切り分け実現方法をまとめた．特に I/O 処理については，ゲスト OS が I/O デバイスを占有し，直接処理できる PCI Pass-through[4)] を応用することでリアルタイム性を保証するのが適切であると判断した．さらに，その判断を裏付けるために，予備実験を行った．予備実験では，PCI Pass-through がリアルタイム性を保証できるか性能評価を行った．その結果，他のゲスト OS が I/O デバイスの処理を行い，

Vol.2010-SLDM-144 No.31
Vol.2010-EMB-16 No.31
Vol.2010-MBL-53 No.31
Vol.2010-UBI-25 No.31
2010/3/27

管理 OS に負荷を与えるときに PCI Pass-through を採用した場合でも，その性能に揺らぎが生じることが分かった．RT-OS が VM 上で動作するとき，I/O デバイスの性能に揺らぎが生じると，外部からの割込み処理や応答が正確に行えず，リアルタイム性を保証することが困難となるため PCI Pass-through の割込み処理の改善が必要となる．

本論文では，以下，第 2 章でリアルタイム仮想計算機モニタ Natsume の概要と構成について述べ，第 3 章では既存仮想計算機モニタとして Xen の概要と VM 上で RT-OS を動作させるうえでの問題点と PCI Pass-through について述べる．第 4 章では I/O 処理について行った予備実験の評価内容と考察について述べ，第 5 章でリアルタイム性保証のための PCI Pass-through の割込み管理手法について提案し，最後に 6 章で本論文のまとめとする．

2. 仮想計算機モニタ　Xen

2.1 Xen の構成

Xen は，サーバシステムの統合を目的に開発されているオープンソース VMM である．Xen はドメインと呼ばれる 2 種類の VM を提供している．1 つは，実際の I/O を処理したり，VM を管理するための管理 OS が動作する，Domain-0 と呼ばれる特権ドメインである．もう 1 つは，その他の OS が動作する非特権ドメインの Domain-U である．

また，Xen では，CPU やメモリ，I/O デバイスといった計算機資源を抽象化し，仮想的な計算機資源を提供する方式として，準仮想化方式と完全仮想化方式の二つの方式を備えている．Natsume では，より実環境上に近い性能を実現する準仮想化をベースとして用いている．

2.2 Xen における資源割当て

Xen では，複数ドメインに対する，計算機資源の割当てを図 1 のような構成で行っている．以下，CPU，I/O，メモリの管理方法について述べる．

CPU 管理

CPU 資源を各ドメインが持つ仮想 CPU（以下，vcpu）として割り付け，スケジューリング（ドメインスケジューリング）することで，複数ドメインの動作を実現している．ドメインスケジューリングでは，各ゲスト OS に対して公平性が保たれるよう，割当て時間を基に CPU の割当てがなされている．Xen は各 CPU に用意された Run キューに登録されている VCPU に物理 CPU を割り当てる．Xen 上で動作するゲスト OS はアイドル状態に遷移すると，ドメインスケジューリングの対象から外れるように自ら依頼し，次にプロセスが起床するまで待機状態となる．

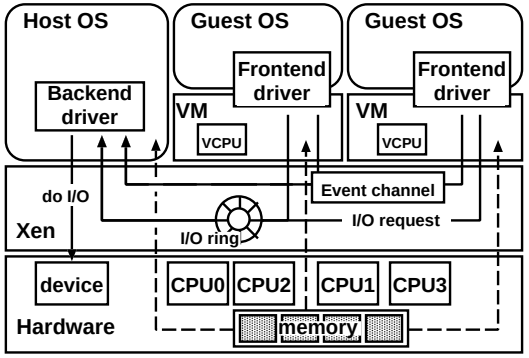


図 1　Xen における資源割当て

I/O 処理

Xen における I/O 処理は，I/O 要求に対して Xen が処理を行うのではなく，Domain-0 上の管理 OS が I/O 要求を代行処理をする．ゲスト OS が I/O 処理を実行する場合，図 1 中の Frontend driver から I/O 要求を I/O リングと呼ばれるキューに格納し，event chanel を通じて処理要求があることを通知する．この通知を受けた管理 OS の Back-end driver がゲスト OS の I/O 処理を代行し，その結果を I/O リングに書込み，event chanel を通じてゲスト OS に通知することで I/O の仮想化を実現している．

メモリ管理

Xen では，メモリはハイパバイザとバルーンドライバによって管理され，各 OS に割り当てられる．物理メモリの容量を超えてメモリを割り当てようとすると，バルーンドライバはゲスト OS の未使用のメモリ領域を回収して割り当てる．

2.3 PCI Pass-through

PCI Pass-through は，Xen 上の特定の VM に対して PCI デバイスを排他的に占有させることで，ゲスト OS による物理 PCI デバイスを直接制御を可能にする機構で，Xen 3.2.0 以降で実装されている．この PCI Pass-through によって従来の Xen における I/O モデルよりも，実環境上に近い性能を実現することが可能となる．

ゲスト OS が I/O 要求を行う際は，I/O 命令によって直接 I/O デバイスを操作することができる．また DMA 転送を用いる場合には，ハードウェアの仮想化支援機構である．Intel Virtualization Technology for Directed I/O（以下，VT-d）によって提供される IOMMU がゲスト OS で指定した転送先アドレスを実際のアドレスに変換することで，ゲスト OS 上

Vol.2010-SLDM-144 No.31
Vol.2010-EMB-16 No.31
Vol.2010-MBL-53 No.31
Vol.2010-UBI-25 No.31
2010/3/27

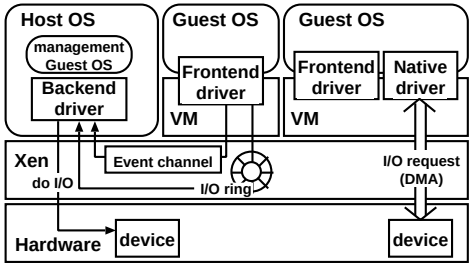


図 2 PCI Pass-Through を適用したときの I/O 処理

からの Native Driver での直接処理を実現している．

3. リアルタイム仮想計算機モニタ Natsume

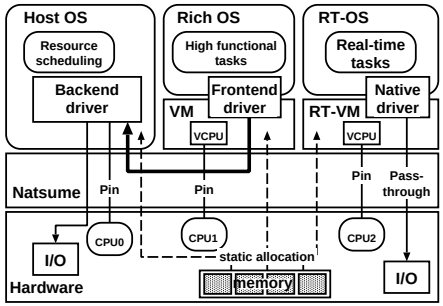


図 3 RT-VMM の構成

3.1 目　　的

従来の組込みシステムでは，リアルタイム性の高い処理と高い機能性を実現するために，(1)RT-OS へ高機能な処理を追加する方法，(2) 高機能 OS へリアルタイム処理機能を追加する手法，(3) 一つの製品内に処理内容ごとに独立した複数の計算機を搭載する手法のいずれかが採られてきた．しかしながら，(1) と (2) の手法では，RT-OS と高機能 OS の性質上の違いから，リアルタイム性を保証することは困難であり，組込みソフトウェア開発による開発コストを増加させる．また，(3) の手法では，一つの製品内に計算機を複数搭載することとなり，製品サイズの拡大や，計算機間の配線複雑化，使用電力の増加，製品単価の増加を招く．

この様な背景から，単一の計算機上で RT-OS と高機能 OS の共存動作を実現するシステムソフトウェア，RT-VMM Natsume（以下，Natsume）を Xen をベースとして開発することとした．Natsume は，図 3 に示すように，RT-OS が動作する RT-VM と，高機能 OS が動作する VM の 2 種類の VM を提供する．これにより，製品サイズや製品コストの抑制，物理的に同じ計算機上であることから，OS 間の配線の複雑化が解消される．また，機能ごとに OS を分割して開発することで，OS 単位での既存システムの流用が可能になる．

3.2 Natsume 実現のための計算機資源の固定割当て

Natsume 実現における課題点と Natsume で行っている RT-VM と Domain-0 に I/O デバイスと CPU コア，メモリなどの各計算機資源を固定割当てについて述べる．

CPU スケジューリング

VMM は複数のゲスト OS に対して仮想化された計算機資源を提供するため，ゲスト OS に対して CPU を時分割でスケジューリングして割り付けている．Natsume のベースとなる Xen では，CPU の割当て時間を公平にするクレジットスケジューラ方式を用いており，ゲスト OS の処理内容やリアルタイム性の保証を考慮したスケジューリングをしていない．すなわち，ゲスト OS が割込み応答性能と最悪処理時間を保証できない恐れがある．この問題を回避するために，Natsume では VCPU と CPU を 1 対 1 で CPU を固定的に割り当てる．これにより，CPU の状態遷移が起きず，CPU 遷移にかかるオーバヘッドを解消し，処理時間のゆらぎ抑制や発生した割込みの即時実行が可能となる．

I/O モデル

Xen では，ゲスト OS の I/O 処理は，VMM を経由して，物理デバイスを管理しているドライバドメインへ通知され，そこで一括して行われる．そのため，I/O 処理要求にオーバヘッドやゆらぎが生じる．また，Xen ではクレジットスケジューラが用いられるため，I/O 処理要求の多いデバイスが存在すると，ドライバドメインが頻繁に起動することになり，ドライバドメインに対する CPU 割当てがされにくくなる．この結果として，I/O 処理の遅延を起こしてしまう恐れがある．

この問題を回避するために，Xen3.4 から実装された PCI Pass-through を用いて，本来，VMM が管理する I/O デバイスを各ゲスト OS に直接割り当てることができるようになった．これによって，ゲスト OS が I/O デバイスを直接使用することで，I/O 処理実行時のオーバヘッドを削減することができる．また，各ゲスト OS が独立して I/O デバイスの処理を行うことにより，Xen の I/O モデルによる I/O 処理のオーバヘッドやゆらぎを回避できる．4 章において，この PCI Pass-through について，RT-VM 上

Vol.2010-SLDM-144 No.31
Vol.2010-EMB-16 No.31
Vol.2010-MBL-53 No.31
Vol.2010-UBI-25 No.31
2010/3/27

の RT-OS が提供するリアルタイム性の保証において問題がないか検証する目的で従来の I/O の処理機構と PCI Pass-through, 実環境上の I/O 処理について性能評価を行う．

メモリ

Xen では，ページング等によるメモリ仮想化は行われていないため，リアルタイム性については特に問題ない．

4. PCI Pass-through におけるリアルタイム性能評価

4.1 計測環境

Natsume を実現する上で必要となる，I/O デバイスを固定割当てを行う，PCI Pass-through を適用したときの I/O 性能について，VM 上で動作する RT-OS のリアルタイム処理に問題がないか，また，他のゲスト OS の動作に影響されないか，計測・評価を行った．

表 1 PCI Pass-through の評価環境

Mother Board	Intel DX58SO
CPU	Intel(R) Core(TM) i7 920(D0 Step) 2.67GHz
memory	DDR3-1333(PC3-10600) 2GB x 3
NIC (eth0)	Gigabit Network Connection 82567LM-2
NIC (eth1)	82541GI Gigabit Ethernet Controller

今回，性能評価をするにあたって利用した評価環境を表 1 に示す．計測端末は同じ構成の端末を 3 台用意し，図 4 のように接続した．

OS には最小構成の debian 5.0 linux(2.6.26-2-686) 32bit を使用し，Xen には Xen 3.4.1(2.6.18.8-xen) を使用した．また，各ゲスト OS に対してメモリは 512MB 使用し，スワップはオフにしている．2 台の端末間を ethernet のクロスケーブルを用いて接続し，スループットを計測した．計測にはメモリ上のデータを転送し，それに要した時間を基にスループットを計測する netperf を用いた．今回の計測では TCP のスループットを 50 回計測し，その平均を基に評価した．また同時に，割込み回数も計測した．負荷プログラムには指定した IP アドレスに対して空の UDP パケットを連続して送りつづけるプログラムを作成し，使用した．

次に今回行った計測のパターンについて述べる．今回行った計測は同種類のデバイスを他のゲストが使用していたときの処理の影響について計測する目的で行った．具体的な計測パターンは図 5 に示すように，3 通りの構成になっている．

case 1

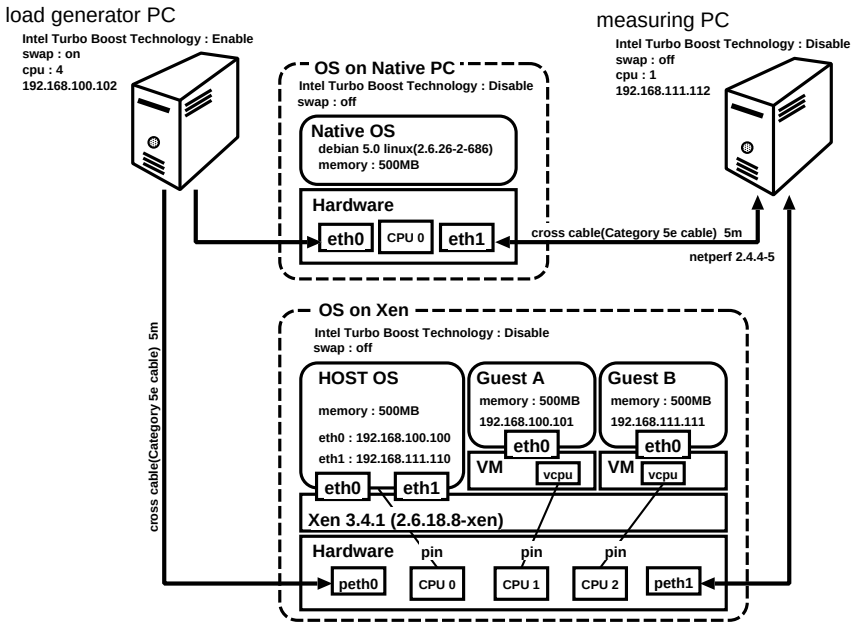


図 4 計測の環境

case 1 は，実環境上での計測である．実環境上の OS が NIC を 2 つ持ち，eth1 で I/O 処理の計測を行う．さらに，eth0 に接続された負荷生成端末から大量の UDP パケットを受信した時の eth1 の I/O 処理性能も計測した．この環境では case 2，case 3 の評価対象であるゲスト OS と合わせるため，評価対象端末の CPU は 1 つだけ有効にしている．

case 2

case 2 は，VM 上のゲスト OS が Xen が提供する仮想 NIC デバイス (以下，vif) を経由して処理する場合の計測である．この計測では，各ゲスト OS に提供する vif は，管理 OS を通じて異なる NIC(以下，peth) に割り付けられている．これは，両ゲスト OS の I/O 処理が実デバイスが独立していてもどれだけ互いに影響しあうかを計測することを目的とする．計測にはゲスト B を I/O 処理性能を計測するゲスト OS に，ゲスト A を UDP を大量に受けるゲスト OS とした．なお，CPU 割当ては各ゲスト OS の VCPU に対して固定的に CPU を一つ割り当てた．

Vol.2010-SLDM-144 No.31
Vol.2010-EMB-16 No.31
Vol.2010-MBL-53 No.31
Vol.2010-UBI-25 No.31
2010/3/27

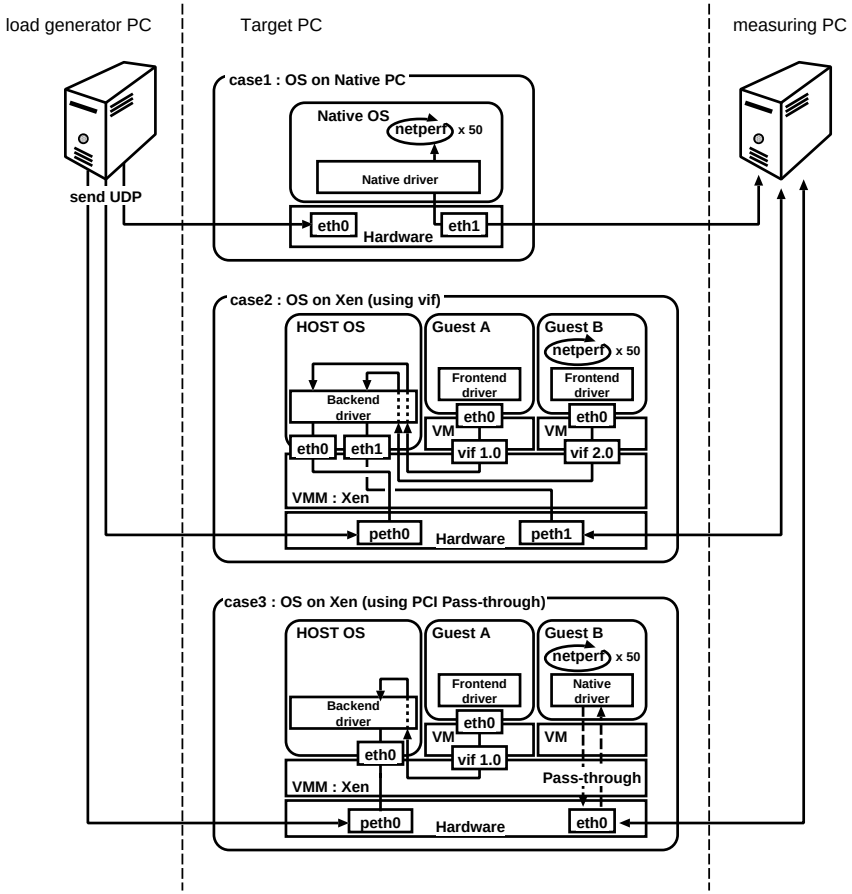
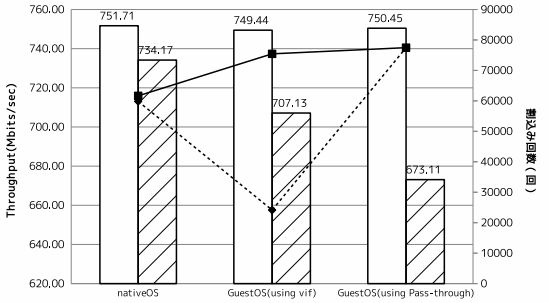


図 5 計測のパターン

case 3

case 3 は，VM 上のゲスト OS が PCI Pass-through を用いて直接 I/O デバイスを扱う場合の計測である．この計測ではゲスト A には case 2 と同様に vif 経由で peth0 を割り当てている．一方で，ゲスト B には PCI Pass-through を用いて直接 I/O デバイスを割当てている．この構成においてゲスト A が UDP パケットを大量に受けることによる負荷が，ゲスト B の I/O 処理にどの程度の影響を及ぼすのかを計測することを目的とする．



throughput (Mbits/sec)	w/o load	751.71	749.44	750.45
	with load	734.17	707.13	673.11
Interrupt (回)	w/o load	61652.41	75445.00	77472.41
	with load	59846.45	24161.04	77448.49

図 6 送信時の計測結果

4.2 評 価 結 果

4.2.1 送信処理の結果

ゲスト OS B から評価用端末への送信時の計測結果を図 6 に示す．グラフは棒グラフが左縦軸と対応しており，ゲスト OS B でのスループット (Mbps/sec) を示している．また，折れ線グラフが右縦軸と対応しており，netperf による計測 1 回にゲスト OS B に通知された割込み回数の平均を示している．

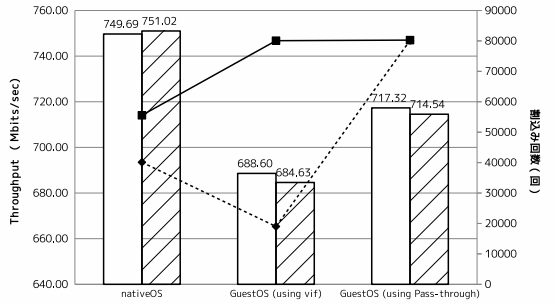
負荷のない環境での処理性能

負荷のない状態におけるスループットは，vif を用いた環境も，PCI Pass-through を用いた環境も，実環境上とほぼ同等の性能が得られることがわかった．また，割込み回数は，vif と PCI Pass-through の両者ともに実環境より多くなる結果となった．

UDP 負荷のある環境での処理性能

vif を利用しているゲスト OS A へ，負荷生成 PC から UDP パケットを送信することによって負荷をかけた状態において，vif が PCI Pass-through の性能を上回る結果となった．また，ゲスト OS A に UDP 負荷をかけた状態では実環境と PCI Pass-through の割込み回数は通常状態と変わらないが，vif 環境では割込みの大幅な減少が確認された．この性能結果は，予想していた結果とは逆の結果であり，vif 環境よりも PCI Pass-through を適用した環境に大きな処理性能のゆらぎが生じることが分かった．これについて，割込み回数は vif が PCI Pass-through を下回ることから，vif では負荷

Vol.2010-SLDM-144 No.31
Vol.2010-EMB-16 No.31
Vol.2010-MBL-53 No.31
Vol.2010-UBI-25 No.31
2010/3/27



Throughput (Mbits/sec)	Load free	749.69	688.60	717.32
	UDP send DomU	751.02	684.63	714.54
Interrupt (回)	Load free	55559.10	80070.39	80257.47
	UDP send DomU	40123.06	19007.84	80143.86

図 7 受信時の計測結果

がかかった際に割込み通知の完了を待たずに管理 OS が I/O 処理を行い，その結果をまとめて通知することで，PCI Pass-through よりも高い送信性能を実現していると考えられる．

4.2.2 受信処理の結果

評価用端末からゲスト OS B へ送信したときのゲスト OS の受信時の計測結果を図 7 に示す．図 6 と同様に，グラフは棒グラフが左縦軸と対応しており，ゲスト OS B でのスループット (Mbps/sec) を示している．また，折れ線グラフが右縦軸と対応しており，netperf による計測 1 回にゲスト OS B に通知された割込み回数の平均を示している．

負荷のない環境での処理性能

負荷のない状態においては，実環境，PCI Pass-through，vif の順で高い性能が得られ，PCI Pass-through が実環境の次に高い処理性能を実現できていることが分かった．また，割込み回数は，vif と PCI Pass-through の両者とも同程度となり，実環境より増大するという，送信時の結果と同様の傾向となった．

UDP 負荷のある環境での処理性能

vif を利用しているゲスト OS A に UDP パケットを送信することによって，負荷をかけた状態においては，送信時の結果とは異なり，負荷をかけていない状態とその性能に差があまりないことが分かった．また，割込みについては送信時の計測結果と同様に，負荷がかかっている状況では割込み回数が大幅に減少した．

4.2.3 考　　察

UDP 負荷がかかっていない状態の送信においては，全ての計測環境でスループットの計測結果にあまり差がでず，リアルタイム性の保証に問題がないと言えるが，UDP 負荷をかけた状態では，vif と Pass-through の性能が実環境と比べ，低下する．また，その低下の幅は Pass-through における結果が最大となっている．このことから, vif よりも Pass-through の方が他のゲスト OS の処理に影響され，他のゲスト OS の処理内容によって I/O 処理に大きなゆらぎが生じ，リアルタイム性を保証できない可能性がある．

割込み回数について，UDP 負荷を与えた環境下で vif 環境よりも PCI Pass-through を適用した環境に大きな処理性能のゆらぎが生じることが分かった．これについて，割込み回数は vif が PCI Pass-through を下回ることから，vif では負荷がかかった際に割込み通知の完了を待たずに管理 OS が I/O 処理を行い，その結果をまとめて通知することで，PCI Pass-through よりも高い送信性能を実現していると考えられる．

最後に，送信処理において PCI Pass-through が vif の性能に劣る理由について考える．PCI Pass-through は，ゲスト OS のデバイスドライバからの I/O 処理である in や out 命令は直接実行できるが，そのデバイスからの割込み処理は従来どおり Xen を介して行われる．そのため，vif と同様に他のゲスト OS によって，管理 OS がビジー状態になった場合，その影響を受ける．しかし，vif は上記のような理由により処理性能が向上していると考えられるが，PCI Pass-through を用いることでゲスト OS 自体はビジー状態にならないため，割込み処理における影響を受けてしまうと考えられる．そのため，この PCI Pass-through の割込みを直接，ゲスト OS が受けるような機構を用意しなければならない．

5. IRQ の共有とリアルタイム性能の評価

5.1 Xen の割込み処理

前節の結果から，Xen 内部で PCI Pass-through が適用されたときの I/O 割込みの処理の流れを確認する目的で，Xen のゲストへの割込み通知の流れについて詳細に調査した．

PCI Pass-through が適用された I/O 割込みは，他の割込み同様に do_IRQ 関数を通り，__do_IRQ_guest によって図 8 のように，割込み番号に対して登録されているゲスト OS へソフトウェア割込みとして通知される．対象となるゲスト OS の数は nr_guest で示されている．具体的にはドメイン構造体の guest を指定して，send_guest_pirq を呼び出すことでソフトウェア割込みを通知させている．すなわち，発生した割込み番号に対して，複数のデバイスが登録されている場合，その登録されている全てのゲスト OS に対して割込みを通知

Vol.2010-SLDM-144 No.31
Vol.2010-EMB-16 No.31
Vol.2010-MBL-53 No.31
Vol.2010-UBI-25 No.31
2010/3/27

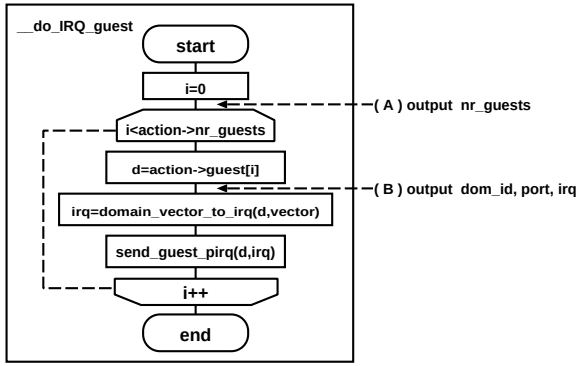


図 8 __do_IRQ_guest 内の処理

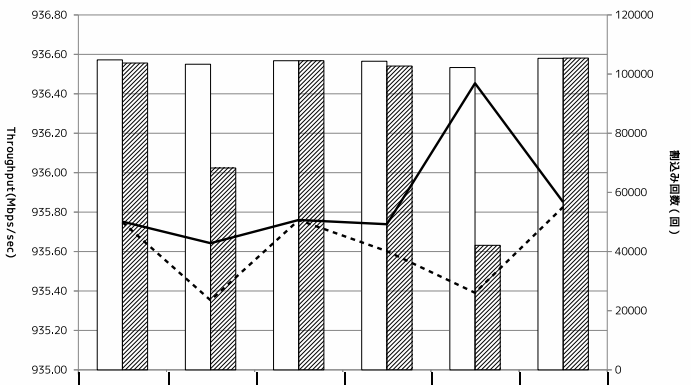
するようになっている．

以上の調査に基づき，その動作を確認するために図 8 の関数内で (A) において発生した割込みに登録されているゲスト OS の数 (nr_guests) を，(B) において送り先のドメイン ID と port 番号，割込み番号を出力した．この結果，PCI Pass-through を適用しているデバイスからの割込みであっても，1 つの割込み番号に対して複数のデバイスが登録されている場合があることがわかった．すなわち，割込み番号を共有するデバイスを他のゲストが使用しているのであれば，PCI Pass-through を適用したデバイスの割当て先ゲスト以外の OS にも通知されることになる．

5.2 IRQ が重複していない PCI デバイスにおける評価

計測端末と計測方法は 4 章の計測とほぼ同様である．ただし，計測に使用する NIC と UDP パケットを受ける NIC が入れ替えた．これによって，計測に使用する NIC は割込み番号を他のデバイスと共有することのない環境とした．

計測結果を図 9 に示す．左縦軸が棒グラフと対応し，スループットを示している．また，右縦軸が折れ線グラフと対応し，1 回の計測における割込み回数を表す．なお，横軸については計測パターンを表している．計測パターンは左 3 パターンがゲスト OS 送信時の結果，右 3 パターンがゲスト OS 受信時の結果となっており，それぞれ図 5 の case 1：実環境，case2：vif，case3：PCI Pass-through の順となっている．今回の計測では，4 章の計測結果とは，割込み番号に登録されているデバイス数だけでなく，デバイスや PCI バスなど異なる点が多いため比較できない．そのため，本計測における各計測パターンの傾向について考察する．スループットの面では，vif 環境において約 0.6Mbps〜1.0Mbps の性能低下が見ら



		nativeOS	GuestOS (vif)	GuestOS (Pass-through)	nativeOS	GuestOS (vif)	GuestOS (Pass-through)
		send			receive		
Throughput (Mbits/sec)	w/o load	936.57	936.55	936.57	936.57	936.53	936.58
	with load	936.56	936.02	936.57	936.54	935.63	936.58
Interrupt (回)	w/o load	50210.02	42830.69	50670.12	49236.16	96899.86	56703.92
	with load	49931.22	23508.57	50643.29	40125.65	26056.20	55081.55

図 9 eth0 を固定的に割り当てた時の計測結果

れたものの，大きな性能差は表れなかった．この結果より，vif と PCI Pass-through 共に性能面では CPU の処理能力が十分であれば，実環境上と同等の性能が出せると言える．一方で割込み回数について，実環境と PCI Pass-through 環境の割込み回数が送受信の両方のパターンにおいてほぼ同値の結果となった．この値によって，4.2 節の PCI Pass-through 環境の割込み回数が実環境よりも平均として多かったのは，割込みを共有している他のデバイスの割込みが発生していたことが原因であると考えられる．また，PCI Pass-through 環境のスループットと割込み回数がほぼ同値であるため，VM 上であっても，PCI Pass-through を使うことで問題なく動作させることが可能であると言える．

5.3 ま と め

以上で述べた二通りの計測結果によって割込みを共有しない PCI Pass-through デバイス，もしくは CPU 能力を十分にもっている環境であれば，VM 上の RT-OS がリアルタイム性を保証する観点から考えたとき，実環境上と同様に，問題なく動作可能であることが確認できた．しかし，vif のように全てのゲスト OS の I/O 処理を管理 OS が代行する仕組みは，RT-OS の処理が管理 OS に依存してしまうことになり，リアルタイム性だけでなく，信頼性の面でも望ましくない．また，割込み番号を共有しているデバイスを使用する PCI

Vol.2010-SLDM-144 No.31
Vol.2010-EMB-16 No.31
Vol.2010-MBL-53 No.31
Vol.2010-UBI-25 No.31
2010/3/27

Pass-through の I/O 割込みについても，RT-OS が実際にはデバイスを使っていないにも関わらず，他の OS の I/O 処理に起因する割込みが通知される．そのため，RT-VM には割込み番号を共有していない I/O デバイスを PCI Pass-through によって割り当てるのが望ましい．しかし，割込み番号を共有する I/O デバイスに対してもリアルタイム性を保証するために，PCI Pass-through の I/O 割込み管理について改善する必要がある．

6. リアルタイム性保証のための PCI Pass-through の割込み管理手法

本章では，IRQ を共有しない I/O デバイスについて通知先 CPU の固定化による PCI Pass-through のリアルタイム性向上と，IRQ を共有するデバイスのリアルタイム性保証のための割込み管理手法について提案する．

6.1 RT-VM が動作する CPU への割込み直接通知手法

Xen では，割込みが発生した際に割込み通知の処理を行う CPU には最も優先度の低い CPU が使用される．このため，Xen では通知するドメインの決定後，通知先ドメインが使用している CPU に対して割込みを発生させることでゲスト OS に通知している．この処理によるゆらぎを抑えるため，占有デバイスの割込みの通知先 CPU を RT-VM が使用している CPU に限定する．具体的には，PCI Pass-through によって占有する I/O デバイスの IRQ に対応する IOAPIC のリダイレクションテーブルを設定し，RT-VM に固定割当てしている CPU に変更することで実現する．これによって RT-VM が占有している I/O デバイスが発生させた割込みは，常に RT-VM に固定割当てされている CPU に通知され，割込みを別のプロセッサに転送するための処理によるゆらぎを抑えることが可能となる．

6.2 IRQ を共有するデバイスの割込み管理手法

5 章において，PCI Pass-through を用いた環境で合っても，図 10 に示すようにデバイスが割込み番号を共有することによって他の OS の I/O 処理の影響を受けることが分かった．PCI Pass-through による RT-VM への I/O デバイス固定割当てにおいて，割込み番号を共有しているデバイスであっても，RT-OS が他の OS の処理に影響を受けることなく動作するにはゲスト OS への割込み通知を割込み番号ではなく，割込みを発生させたデバイスごとに決めるべきである．また，これによって RT-OS の I/O 処理におけるリアルタイム性保証だけでなく，不必要なデバイス割込みを解消することによって高機能 OS の I/O 処理性能向上にもつながる．本手法では，ゲスト OS への割込み通知の判断を割込み番号ではなく，PCI デバイスの BDF(Bus No. Device No. Function No.) 情報によって判断する．

PCI Pass-through を適用する際にデバイスの指定に使われる BDF 情報と割込み番号を用いることで，図 11 のように割込み番号に登録されたゲスト OS 全てに送るのではなく，対象のゲスト OS に直接送ることができる．

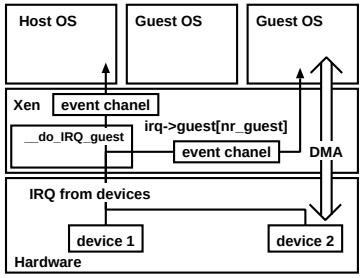


図 10 従来の割込み管理手法

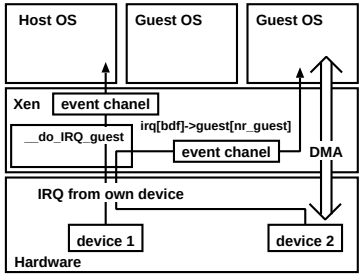


図 11 提案する割込み管理手法

7. おわりに

代表的な仮想計算機モニタのひとつである Xen をベースに，同じ計算機上で高機能な OS がハイエンドな処理をしつつ，リアルタイム OS がリアルタイム性を維持し，共存動作することを可能にするリアルタイム仮想計算機モニタ「Natsume」の開発を行っている．本論文では，特に I/O 処理についての課題を解決することを目的として，ゲスト OS が I/O デバイスを占有し，直接処理できる PCI Pass-through がリアルタイム性を保証できるかの性能評価を行い，その結果を基にしたリアルタイム性を保証するための PCI Pass-through の割込み管理手法を提案した．

参 考 文 献

1) 経済産業省 中小企業庁：中小企業の特定ものづくり基盤技術の高度化に関する指針 (2009).
2) 永井正武監修. 澤田勉. 権藤正樹. 永井正武共著：実用組込み OS 構築技法, 共立出版, pp.6–15 (2001).
3) Barham, P. Dragovic, B. Fraser, K. Hand, S. Harris, T. Ho, A. Neugebauer, R. Pratt, I. Warfield, A. : Xen and the art of virtualization, Proceedings of the nineteenth ACM symposium on Operating systems principles,(2003).
4) Yuji Shimada : Development of the I/O Pass-through Current Status and the Future, Xen Summit Asia 2008,(2008).