Vol.2010-OS-113 No.3
2010/1/27

# XenにおけるPCI Passthroughの性能評価

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

現在，Xen 上のゲスト OS としてリアルタイム OS(RTOS) を動作させることを視野にいれ，Xen のリアルタイム性について検証を行っている．本論文では，そのうち，入出力に関する評価を行ったので，その結果について報告する．具体的には，資源を特定のゲスト OS へ排他的に割り当てる機能である PCI Passthrough の性能について，CPU バウンドな負荷や I/O バウンドな負荷を，ホスト OS やゲスト OS にかけつつ評価した．また，仮想デバイスドライバを用いた場合などについても評価した．

# Performance evaluation of PCI Passthrough of Xen

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

We have been making verification of Xen hypervisor's capability about real time, to run real time operating systems(RTOS) on it. We have experiments of I/O performance in particular. In this paper, we discuss its result and a study. To be concrete, we have evaluated performance of PCI Passthrough, a mechanism to assign devices to a specific guest OS exclusively, in the following situations. First is a situation with CPU-bound load on guest OS and/or Host OS. Second is a situation with I/O-bound load on guest OS and/or Host OS. Furthermore performance of virtual device drivers is also evaluated.

†1 立命館大学情報理工学部
College of Information Science and Engineering, Ritsumeikan University
†2 立命館大学大学院理工学研究科
Graduate School of 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，組込みシステム分野において，ハードウェアとソフトウェアの開発コストの削減や，省電力化によるランニングコストの削減が求められている．加えて，携帯電話や車載システム，FA といったシステムでは，信頼性・応答性に加えて，高い機能性の両立も求められている．このような現状から，組込みシステム分野では，仮想計算機技術に注目が集められている．組込みシステムには，VxWorks や $\mu$ITRON に代表されるリアルタイム OS(以下，RTOS) が利用されており，組込みシステムに仮想計算機モニタ (以下，VMM) を導入することで，複数の RTOS を仮想計算機 (以下，VM) 上で動作させることが可能になる．これにより，システムを構成するハードウェア数を削減でき，ハードウェアの開発コストや省電力化による運営コストの削減を実現することが可能になる．また，RTOS と同時に，Windows や Linux に代表される高い機能性を持った高機能 OS を動作させることにより，RTOS で応答性と信頼性を保証しつつ，高機能 OS によって，ユーザインターフェースやネットワーキングなどの高い機能性を実現することが可能になる．

RTOS は，機器の制御を主要機能とした最小限の機能提供により，処理時間の予測性を高め，高い応答性と信頼性を保ち，そのリアルタイム性能を保証している[1)]．しかし，RTOS を複数のゲスト OS が動作する VMM 上で動作させた場合，物理資源を複数の VM で共有するため，他のゲスト OS や VMM にかかる負荷の影響により，処理性能に変化が発生する場合がある．これは，RTOS が正確な処理時間の予測を行うことを困難にする．つまり，VMM 上で RTOS を動作させる場合，VMM にゲスト OS によるリアルタイム性能の保証を可能にする資源管理機構を実現する必要がある．

そこで，既存の VMM である Xen[2)] 上で，ゲスト OS として RTOS を動作させることを視野に入れ，Xen のリアルタイム性能について検証を行った．仮想化環境におけるゲスト OS 同士の影響を調べるために，クアッドコア CPU や，複数のインターフェースカードやネットワークカードといった物理資源を搭載した計算機を用意し，複数のゲスト OS を動作させつつ，様々な性能評価実験を行った．

本論文では，そのうち，入出力に関する性能評価を行った結果とその考察について報告する．具体的には，物理資源を特定のゲスト OS へ排他的に割り当て，ゲスト OS から直接操作を可能 PCI Passthrough を利用する．この機構により資源を特定の VM に割り当て，外部から CPU バウンドな負荷や I/O バウンドな負荷をかけつつ，ゲスト OS の性能がどのように変化するかを計測し，評価した．また，比較対象として，PCI Passthrough を利

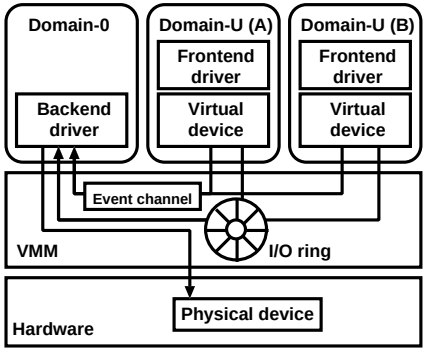


**図 1** 仮想デバイスドライバによるアクセス

用せず，仮想デバイスドライバを用いた場合についても計測・評価を行った．

以下，本論文では，2 章で既存の VMM である Xen の概要と，Xen 上で RTOS を動作させる際の問題点，そして，ゲスト OS に資源を固定的に割り当てる PCI Passthrough について述べる．次に，3 章で入出力に関する Xen のリアルタイム性の評価と考察について述べ，4 章で本論文の内容をまとめる．

## 2. 仮想計算機モニタ Xen

### 2.1 概　　要

Xen は，オープンソースにより開発されている VMM である．Xen は，CPU や I/O デバイスなどの計算機資源を抽象化し，ドメインと呼ばれる仮想計算機環境を提供する方式として，完全仮想化と準仮想化の 2 つを実現している．特に本論文では，物理的に一つの計算機で，複数の仮想環境をネイティブに近いパフォーマンスで実現する準仮想化を対象とする．Xen の準仮想化環境では，Xen 自身の制御や各 VM の管理を行うためのドメインを Domain-0，その他のゲスト OS が動作するドメインを Domain-U と呼ぶ．

### 2.2 RTOS を動作させる上での問題点

Domain-U で動作するゲスト OS は，物理資源を抽象化した仮想資源を利用する際，ネイティブなデバイスドライバの代替として，仮想デバイスドライバを用いる．仮想デバイスドライバは，物理資源と仮想資源を関連付けることで，ゲスト OS 間で物理資源を共有することを可能にする．Domain-0 が持つ仮想デバイスドライバをバックエンドドライバ，Domain-U が持つ仮想デバイスドライバをフロントエンドドライバと呼ぶ．

仮想デバイスドライバを利用する場合，**図 1** に示すように，Domain-U 上からのアクセス要求は，一度 VMM のイベントチャネルや I/O リングを経由し，Domain-0 上で動作する OS(以下，ホスト OS) が処理する．しかし，この方法は，ホスト OS に負荷がかかった場合，要求の完了に遅延が発生する可能性がある．また，抽象化された物理資源は，各ゲスト OS 間で共有されるため，実際の処理性能が他のゲスト OS の動作状況に影響される可能性がある．つまり，仮想デバイスドライバによる資源の割当ては，RTOS による処理時間の正確な予測を困難にし，リアルタイム性を低下させる原因になる．そのため，Xen 上で RTOS を動作させる場合，RTOS が動作するドメインにおいて，処理性能が他のゲスト OS の影響を受けないようにする資源管理機構が必要になる．

以上から，他のゲスト OS の影響を受けにくくするために，物理資源を排他的に RTOS が動作するドメインに割り当てる方法を採る．

### 2.3 物理 CPU の排他的割り当て

Xen では，物理 CPU や物理 CPU コアといった CPU 資源を仮想 CPU として各ドメインに割り当てる．この際，仮想 CPU は，処理を行う CPU 資源が紐づけされた状態で管理され，VMM 内のドメインスケジューラ[3] によって，実際に処理を行う CPU 資源が決定される．この仮想 CPU と CPU 資源の紐づけは，Xen に付属している xm コマンドや Xen Tools といったツール群[4] で変更が可能である．このツール群は，Domain-0 上で動作するホスト OS から利用可能であり，単体の仮想 CPU に複数の CPU 資源を紐づけしたり，逆に複数の仮想 CPU にひとつの CPU 資源を紐づけることが可能である．また，CPU 資源の紐づけの他に，メモリの割当て量の調整やドメインそのものの起動などを行うことが可能である．このツール群を利用し，仮想 CPU と CPU 資源を 1 対 1 で紐づけた上で，ドメインに割り当てることで，CPU 資源の排他的に割り当てることが可能である．

### 2.4 PCI Passthrough

PCI Passthrough は，Xen 上で動作する特定のドメインに対して，PCI デバイスを排他的に割り当てる機構であり，Xen 3.2.0 以降において実装されている．PCI Passthrough は，Intel Virtualization Technology(VT-d)[5] や AMD IOMMU[6] に代表されるハードウェアの仮想化支援機構によって提供される IOMMU を利用することで実現され，PCI Passthrough が適用されたデバイスは，Domain-0 や他の Domain-U から隠蔽される．隠蔽されたデバイスは，PCI Passthrough によって割り当てられた Domain-U 上のゲスト OS による占有が可能になる[7]．また，この機構によって割り当てられたデバイスは，ゲスト OS からネイティブなデバイスドライバを利用してアクセスすることが可能であり，I/O 処理で Domain-0 の

ドライバを利用することなく，直接アクセスすることが可能になる．

PCI Passthrough を用いることによって，他のゲスト OS や VMM の負荷状況や，割込み禁止などの影響を受けることなく，デバイスへの入出力性能を一定に保つことが可能であれば，Xen 上で動作する RTOS のリアルタイム性の保証に有効であると考え，PCI Passthrough を利用した入出力性能の評価実験を行った．

## 3. 入出力性能の評価

### 3.1 実験環境

PCI Passthrough が他のドメインの処理にどれだけ影響を受けるのかを検証するため，Domain-0 や同時に動作する他の Domain-U に対して，CPU バウンドな負荷や I/O バウンドな負荷をかけつつ，Domain-U における入出力性能を計測した．

実験環境を**表 1** に示す．性能評価の対象である評価端末，I/O 負荷をかけるための負荷用端末，そして，ネットワークのトラフィックを計測するための計測用端末をそれぞれ同ハードウェア構成とした．なお，端末間のネットワークは，クロスケーブルを用いて直接接続している．なお，VMM やゲスト OS の性能に影響を与えないように，Intel Speed Step Technorogy[10] に代表されるハードウェアレベルでの省電力機構や，Intel Turbo Boost Technorogy[11] や Intel Hyper-Threading Technology[12] に代表される CPU 性能を動的に変化させる機構は，全て無効にしている．また，今回の実験では，HDD 性能を測定するために，ファイルやデバイスに任意のブロックサイズで入出力が可能な dd コマンドを用いた．ネットワーク性能の計測には，トラフィック測定ツールである netperf[8] を用いた．CPU 負荷の生成には，stress[9] を利用し，I/O 負荷の生成には，任意のサイズの UDP パケットを任意のスレッド数で発生させるプログラムを作成し，これを用いた．

**表 1** 実験環境の構成

| 構成要素 | 型番 |
|---|---|
| CPU/Motherboard | Intel Core i7 920 2.67GHz/Intel DX58SO |
| NIC1(eth0) | 82567LM-2 Gigabit Network Connection |
| NIC2(eth1) | 82541GI Gigabit Ethernet Controller |
| SATA I/F | REX-PE30S (SiI3132) |
| HDD | HDP725050GLA360 (SATAII, 500GB, 7200rpm) |
| VMM | Xen 3.4.1 |
| OS(Native) | Debian GNU/Linux 5.0 (kernel 2.6.26-2) |
| OS(VM) | Debian GNU/Linux 5.0 (kernel 2.6.18-xen) |
| HDD 性能測定ツール | dd 6.10 |
| トラフィック測定ツール | netperf 2.4.4-5 |
| CPU 負荷発生ツール | stress 0.18.9 |
| I/O 負荷発生ツール | udpsend（自製） |

### 3.2 ディスク I/O 性能の評価

#### 3.2.1 CPU 負荷をかけた計測の方法

CPU 負荷をかけた状態でディスク I/O の性能を計測する方法について説明する．この実験は，同時に動作している他のドメインにおける CPU 負荷の影響を明らかにする目的で行った．具体的には**図 2** と**図 3** に示すように，Domain-0 に CPU コアを 2 つ占有させ，4GB の物理メモリを割り当てた．そして，2 つの Domain-U(A) と (B) に CPU コアを 1 つずつ占有させ，物理メモリを 1GB ずつ割り当てた．Domain-U(A) は，負荷生成専用のドメインであり，Domain-U(B) は，I/O 性能を計測することを目的としたドメインである．計測では，CPU バウンドな負荷を次の 3 つのパターンで生成した．

- ホスト OS 上のみで生成
- Domain-U(A) 上のみで生成
- ホスト OS と Domain-U(A) の両方で生成

さらに，上記の 3 パターンのそれぞれについて，Domain-U(B) への HDD の割当て方式を次の 2 パターンで変化させた．すなわち，6 種類を計測した．

- 仮想デバイスドライバを経由した割り当て
- PCIPassthrough により，SATA カードごとの割り当て

具体的な計測では，dd コマンドを用いた．ただし，オプションで，アクセス対象を割り当てた HDD のデバイスファイル，ブロックサイズを 128KB，カウント数を 8000 と指定し，ファイルシステムを介さず，HDD の先頭 1GB 分のデータを読み込むのに要した時間を計測した．可能な限り種々のディスクキャッシュの影響を除外するために，最も多くの物理メモリを割り当てたドメインの物理資源の容量に相当する，4GB の評価と関連しないデータを読み込ませた後に，dd コマンドを実行した．

#### 3.2.2 I/O 負荷をかけた計測の方法

CPU 負荷をかけた状態でディスク I/O の性能を計測する方法について説明する．この実験

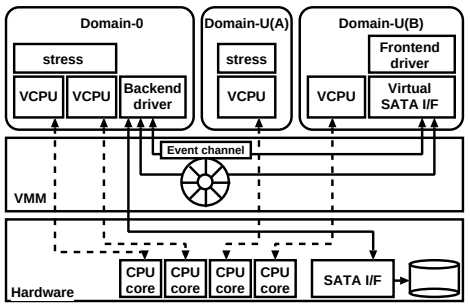


**図 2** CPU 負荷をかけた仮想デバイスドライバの計測

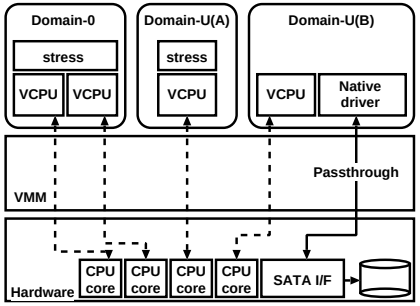


**図 3** CPU 負荷をかけた PCI Passthrough の計測

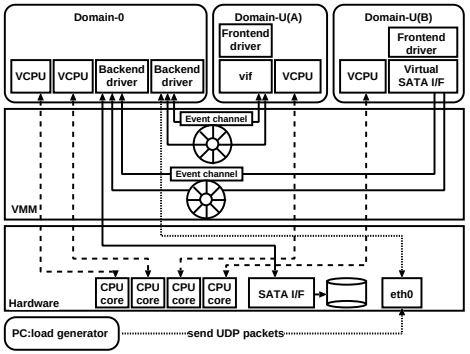


**図 4** I/O 負荷をかけた仮想デバイスドライバの計測

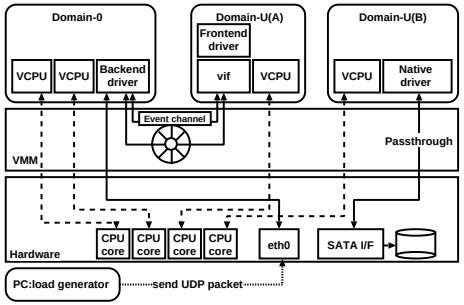


**図 5** I/O 負荷をかけた PCI Passthrough の計測

は，計測対象と種類の異なるデバイスによる I/O 負荷の影響について計測する目的で行った．具体的には，**図 4** と**図 5** に示すように，CPU 負荷を計測した実験と同様に CPU コアと物理メモリを割り当てた Domain-0，Domain-U(A) と (B) を用意し，Domain-U(A) に eth0 に接続した仮想 NIC(以下，vif) を割り当てる．そして，Domain-U(A) に対して外部の負荷用端末から UDP パケットを送り続けることで，I/O 負荷を生成した．また，Domain-U(B) への HDD 割当て方式を，次のパターンに変化させた．

- 仮想デバイスドライバを経由した割り当て
- PCI Passthrough により，SATA カードごとの割り当て

具体的な計測でも，同様に dd コマンドを用い，種々のディスクキャッシュの影響を除外しつつ，先頭 1GB 分のデータを読み込むのに要した時間を計測した．

### 3.2.3 計測結果と考察

以上で述べた計測の結果を，まとめて**図 6** から**図 9** に示す．図 6 は，各計測において，50 回計測した値の平均を表したものである．また，Linux をインストールしたベアマシン環境 (以下，Native 環境) との平均値の差を表したものを**図 7** に示す．図 7 の (10) より，仮想デバイスドライバ経由でアクセスし，且つ I/O バウンドな負荷をかけた場合，Native 環境と比較して遅延は最大 2.82sec 程度となり，大きな性能低下が見られる．また (7) より，CPU バウンドな負荷をかけた場合において，遅延が最大 0.35sec 程度と，I/O バウンドな負荷をかけた場合ほど大きくはないものの，性能低下が見られる．また (4)(5) より，PCI Passthrough にて割り当てたデバイスに対するアクセスの場合，I/O バウンドな負荷と CPU バウンドな負荷で共に最大 0.33sec 程度となり，性能低下は発生するが，仮想デバイスドライバほど大きく性能は低下せず，負荷の種類に関係なくほぼ一定の性能を維持しているという結果になった．ただし，図 6 の (2) より，Domain-0 環境において，アクセスに要する時間が約 60%になり，性能が向上するという結果が得られたが，この原因は現在調査中である．

計測結果から求めた分散を**図 8** に示す．図 8 の (3)〜(7) と (8)〜(12) のそれぞれを比較すると，仮想デバイスドライバを利用した場合の方が，PCI Passthrough を利用した場合よりも，性能のゆらぎが大きいことが分かる，特に，(6) と (11)，(7) と (12) より分かるように，CPU 負荷をかけた場合よりも，I/O 負荷をかけた場合の方が，その傾向が顕著である．また (3) と (8) より，他のドメイン上のゲスト OS や，VMM に負荷をかけていない状態では，何らかの負荷がかかっている状態と比較して，性能のゆらぎが大きいという結果が得られた．この原因は未だ解明できてはいないが，Xen におけるドメインスケジューラに原因があるのではないかと考えている．突然のデーモンの動作や割込み発生といった理由により，スリープしていた仮想 CPU がスケジューリングされる際に，大きなオーバヘッドが発生するのではないかと考え，現在調査を進めている．

次に，計測結果の最大値と最小値を図 9 に示す．図 9 の (3) と (5) より，PCI Passthrough を利用した場合の計測では，最大 50〜96msec 程度の性能のゆらぎが発生した．一方，(10) と (12) より，仮想デバイスドライバを利用した環境における計測では，最大 80〜310msec 程度の性能のゆらぎが発生した．また (1) より，仮想化環境ではない Native 環境では，約 75msec の性能のゆらぎが発生した．すなわち，PCI Passthrough によって，外的な要因による性能のゆらぎを Native により近い状態に抑えられるという結果が観測できた．

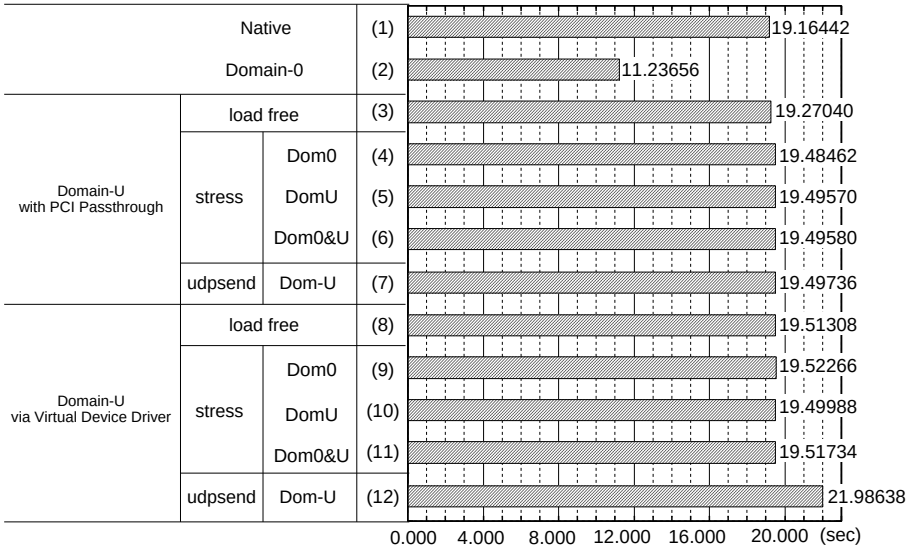


**図 6** dd による計測結果 (平均)

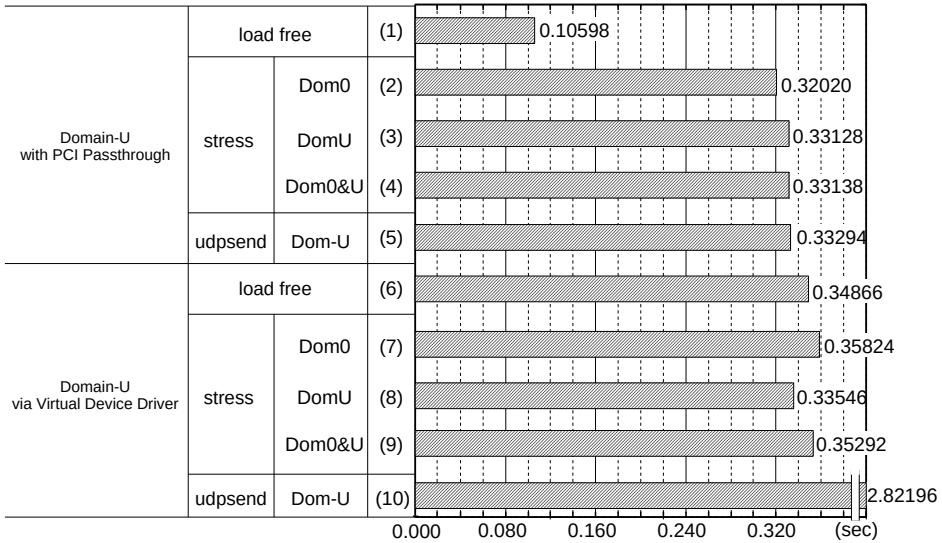


**図 7** dd による計測結果 (Native との平均値の差)

次に，PCI Passthrough によるオーバヘッドについて考察する．Native 環境において 5 回計測を行い，デバイスへの割込みの発生回数を計測したものを**表 2** に示す．これは，IRQ が割り当てられたデバイスの割込みに関する情報を取得可能な /proc/interrupts を利用した．これらの値から算出した平均値と，図 6 の (3)(8) と，図 7 の (2)(6) より，負荷をかけていない状態における，割込み 1 回あたりの処理時間と，Native 環境との差を計算した結果を**表 3** に示す．表 3 から，PCI Passthrough のオーバヘッドによる性能低下は，割込み 1 回あたり約 6.7$\mu$sec であり，割込み 1 回あたりの処理時間の 0.5%程度であることが分かる．なお，仮想デバイスドライバによる性能低下は，割込み 1 回あたり約 22.0$\mu$sec であり，PCI Passthrough と比較して 3 倍程度となっている．加えて，図 9 の (1) から，仮想化によるオーバヘッドのない Native 環境において，最大約 75msec の性能のゆらぎが発生していることが分かる．よって，PCI Passthrough によるオーバヘッドは，Native 環境における性能のゆらぎの 0.009%程度であると計算することができる．すなわち，PCI Passthrough そのもののオーバヘッドは，仮想デバイスドライバと比較して小さく，かつ Native における Linux 環境で発生する性能のゆらぎよりも小さい．これまでの評価を合わせて考察すると，PCI Passthrough は，従来の仮想デバイスドライバと比較して，リアルタイム性を保証するために有効であると考えられる．

### 3.3 ネットワーク性能の評価

#### 3.3.1 計 測 方 法

この実験は，同種類のデバイスを他のドメインで使用している場合の影響を明らかにする目的で行った．具体的には**図 10** や**図 11** に示すように，Domain-0，Domain-U(A) と (B) の 3 つのドメインを用意し，それぞれに CPU コアを 1 つずつ占有させ，物理メモリを Domain-0 に 5GB，Domain-U にそれぞれ 512MB 割り当てた．さらに，Domain-U(A) に eth0 を接続した vif を割り当てた状態で計測を行った．ディスクアクセス性能の計測時と同様に，Domain-U(A) は，負荷生成専用のドメインであり，Domain-U(B) は，I/O 性能の計測を目的としたドメインである．そして，Domain-U(A) に対して，外部の負荷用端末から UDP パケットを送り続けて，I/O 負荷を生成した．さらに，次に示すように，Domain-U(B) への eth1 の割当て方を変化させて，計測した．

- 仮想デバイスドライバを経由した割り当て (vif)

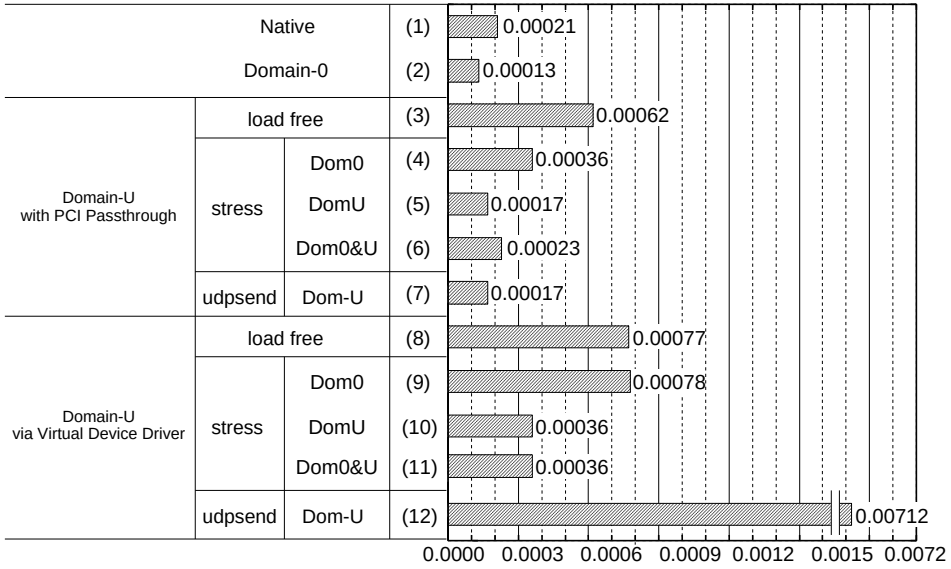


**図 8** dd による計測結果 (分散)

- PCI Passthrough によるコントローラごとの割り当て

具体的な計測では，netperf を用いた．Domain-U(B) で，netperf のデーモンを起動させ，計測用端末で netperf を実行し，受信時のスループットを計測した．また，netperf のデーモンを起動させた計測用端末に対して，Domain-U(B) で，netperf を実行し，送信時のスループットも計測した．

### 3.3.2 計測結果と考察

計測結果を**図 12** に示す．この結果は，それぞれの状況において 5 回ずつ計測した結果の平均をとったものである．まず図 12 の (2) と (3) や (5) と (6) より，負荷なしの状態において，vif を利用した場合よりも，PCI Passthrough の方が，スループットが高かった．これは，特に送信よりも受信の方がその傾向が顕著であるという結果になった．一方，負荷ありの場合，受信時においては負荷なしの場合と同様の傾向が見られた．しかし (3)(9) より，送信時には PCI Passthrough を利用した場合の性能が，負荷なしの場合と比較して，約 50Mbps 減と，明らかな性能低下が見られた．(2)(8) より，vif を利用した場合の性能も，負荷をかけていない状態と比較してある程度低下しているが，約 30Mbps 減と，PCI

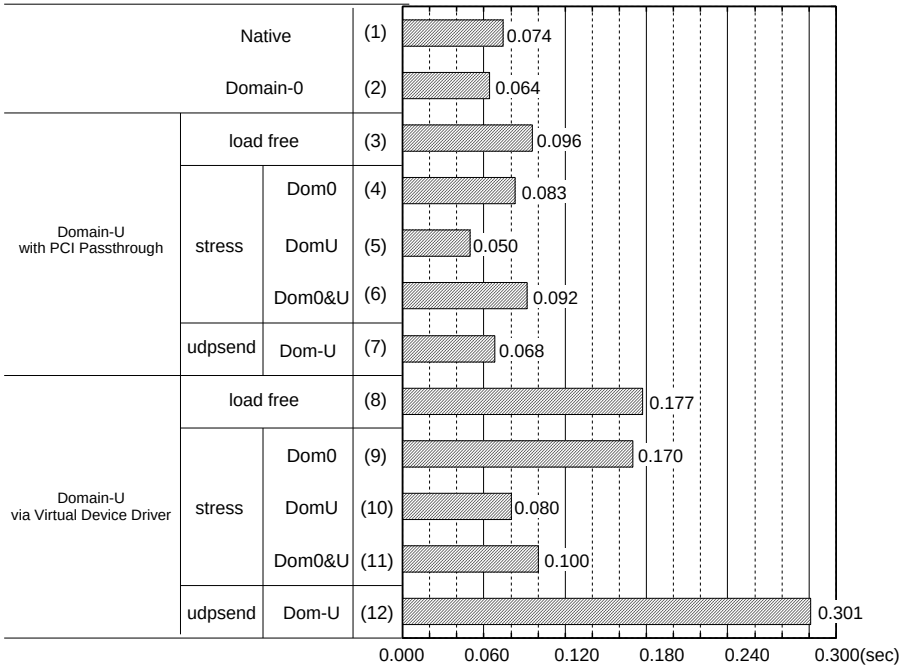


**図 9** dd による計測結果 (最大最小値の差)

**表 2** 発生した割込み回数

| 試行回数 | 1 回目 | 2 回目 | 3 回目 | 4 回目 | 5 回目 | 平均 |
|---|---|---|---|---|---|---|
| 割込み発生数 | 15910 | 16064 | 15997 | 15882 | 15421 | 15854.8 |

**表 3** 1 割込み毎の処理時間

| 割当て | 処理時間 ($\mu$sec) | Native との差 ($\mu$sec) | 割合 |
|---|---|---|---|
| PCI Passthrough | 1215.43 | 6.68 | 0.54% |
| 仮想デバイスドライバ | 1230.73 | 21.99 | 1.78% |

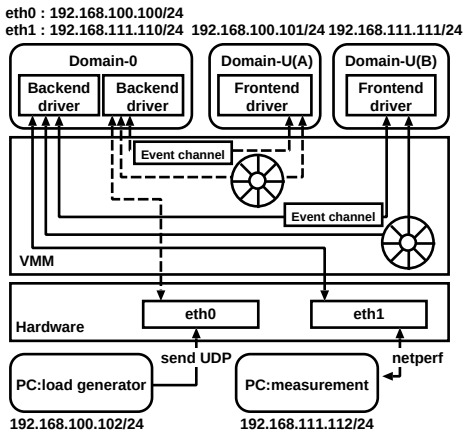


**図 10** 仮想デバイスドライバを用いた NIC の性能計測

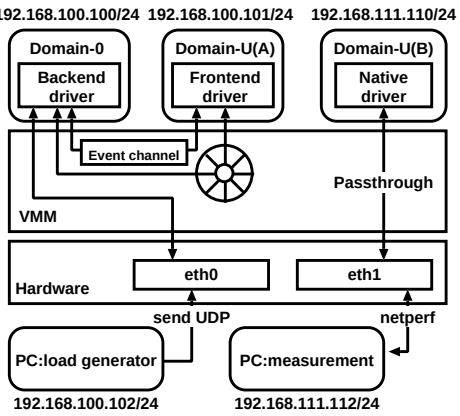


**図 11** PCI Passthrough を用いた NIC の性能計測

Passthrough のそれよりも低下の度合いは少なかった．

このような結果になった原因についてはまだ不明であるが，PCI Passthrough の割込みがホスト OS を経由して伝えられることが原因ではないかと考えている．vif を用いた I/O 処理は，ゲスト OS がホスト OS に対して，処理を I/O リングとイベントチャネルを通じて依頼し，ホスト OS がこれを処理した上で，その結果を I/O リングにより送信するという手法がとられている．これに対して PCI Passthrough では，ゲスト OS が直接データの入出力処理を行うことになるが，割込み通知に限ってはイベントチャネル経由で行われる．そのため，入出力処理中に割込みが発生する頻度の高いデバイスの場合，PCI Passthrough は，割込みが発生する度に制御が VMM やホスト OS にスイッチしてしまうため，ある程度のデータをホスト OS 上でまとめて処理できる仮想デバイスドライバを用いる場合よりも，性能が低下する可能性があると考えられる．つまり，現状の PCI Passthrough は全てのデバイスにおいて必ずしも，リアルタイム性の保証に有効ではなく，割込みが頻繁に発生するデバイスの割当てを行う場合においては，割込み通知をイベントチャネルを介さずに直接行う機構を実装する必要がある．

なお，HDD の性能評価の際に，このような傾向が見られなかった理由は，割込み回数が NIC の場合と比較して少ないためだと考えている．HDD の性能評価と同様に，/proc/interrupts を利用して割込み回数を計測したものを**表 4** に示す．この結果から，PCI Passthrough を

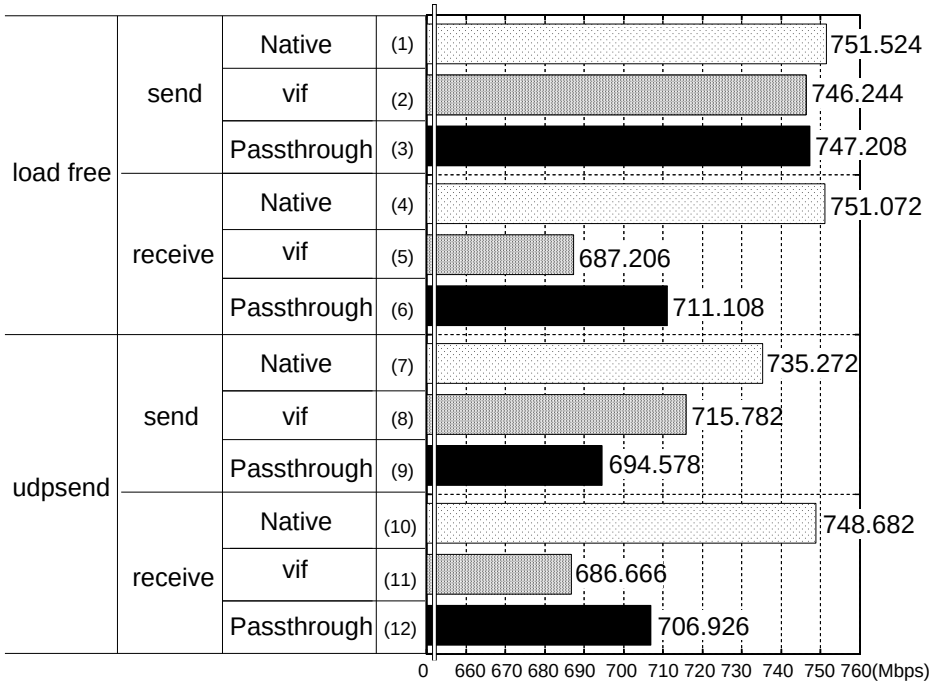


**図 12** netperf による計測結果 (平均)

利用した場合における，送信時の性能評価では，HDD の性能評価とは異なり，約 5 倍の割込みが発生していることがわかる．しかし，受信時の性能評価において割込みの発生回数に大きなゆらぎが見られたことから，割込み回数のみが影響しているとは考えにくい．これを踏まえた結果，一つの計算機で完結する I/O 処理と，ネットワークを介した複数の計算機間の I/O 処理という性質の違いも，影響していると考えている．今回，NIC の測定において利用した TCP は，負荷やノイズなど，何らかの原因によりパケットロスが発生した場合，確実な通信を行うために，パケットの再送を行う．これは，処理のやり直しとなり大きな遅延となるが，HDD へアクセスを行う場合，TCP における再送に相当する，処理のやり直しは発生せず，単純な処理の遅れとなって現れるため，NIC における性能評価ほど大きな影響にならなかったためではないかと考えている．

また，今回計測した受信時における割込み回数のゆらぎの，実際のスループットへの影響

**表 4** netperf における割込み発生回数

| 測定回数 | 割込み発生回数 | | | |
|---|---|---|---|---|
| | vif | | PCI Passthrough | |
| | 送信 | 受信 | 送信 | 受信 |
| 1 回目 | 75720 | 65076 | 77825 | 81534 |
| 2 回目 | 75714 | 69223 | 77254 | 47264 |
| 3 回目 | 75491 | 60738 | 77295 | 51989 |
| 4 回目 | 74964 | 44605 | 77494 | 67101 |
| 5 回目 | 75788 | 70415 | 77509 | 70004 |
| 6 回目 | 76017 | 76833 | 78012 | 90088 |
| 7 回目 | 75642 | 61345 | 77431 | 57656 |
| 平均 | 75619.43 | 64033.57 | 77545.71 | 66519.43 |

については，現在，調査中である．

## 4. おわりに

本論文では，Xen 上で，RTOS を動作させることを視野に入れ，Xen 上で動作するゲスト OS における入出力性能の評価を行った．その際，I/O デバイスを Xen 従来の方式である仮想デバイスドライバを利用して割り当てた場合と，排他的にゲスト OS に割り当てることを可能にする PCI Passthrough を利用して割り当てた場合において，CPU 負荷や I/O 負荷をかけてその性能を測定した．その結果，PCI Passthrough は，CPU 負荷，I/O 負荷ともに性能低下や性能のゆらぎを抑制する効果があり，仮想デバイスドライバと比較してリアルタイム性の保証に少なからず有効であるということが分かった．しかしながら，デバイスによっては，割込み通知のオーバヘッドにより，性能が低下する場合があることも分かった．今後，今回の実験において結果が予測と異なった箇所における原因や割込み回数と実際の性能との関係の追求と，PCI Passthrough を基としたリアルタイム性を保証する機構について，さらに研究を進めていきたい．

## 参 考 文 献

1) 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).
2) Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, Andrew Warfield: *Xen and the Art of Virtualization*, ACM Symposium on Operating Systems Principles (2003).
3) David Chisnall:*The Definitive Guide to the Xen Hypervisor* (2007).
4) William von Hagen:*Professional Xen Virtualization*, Wiley Pubnlishing, Inc, pp. 340-381 (2008) pp217-235.
5) Intel Corporation:*Intel Virtualization Technology for Directed I/O Architecture Specification*, `http://download.intel.com/technology/computing/vptech/Intel(R)_VT_for_Direct_IO.pdf` (2008).
6) Advanced Micro Devices, Inc:*AMD I/O Virtualization Technology (IOMMU) Specification License Agreement*, `http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/34434.pdf` (2009).
7) William von Hagen:*Professional Xen Virtualization*, Wiley Pubnlishing, Inc, pp. 109-115 (2008).
8) Rick Jones:*The Temporary Netperf Homepage*, `http://www.netperf.org/netperf/` (2009).
9) Amos Waterland:*stress project page*, `http://weather.ou.edu/~apw/projects/stress/` (2009).
10) Intel Corporation:*Intel Core i7-900 Desktop Processor Extreme Edition Series and Intel Core i7-900 Desktop Processor Series Datasheet, Volume 1*, `http://download.intel.com/design/processor/datashts/320834.pdf`, p10, p87 (2009).
11) Intel Corporation:*Intel Turbo Boost Technology in Intel Core Microarchitecture (Nehalem) Based Processors*, `http://download.intel.com/design/processor/applnots/320354.pdf` (2008).
12) Intel Corporation:*Intel Hyper-Threading Technology*, `http://www.intel.com/technology/platform-technology/hyper-threading/index.htm` (2009).