Vol.2011-ARC-195 No.13
Vol.2011-OS-117 No.13
2011/4/13

# 仮想計算機モニタ・バイパス型ネットワークに対する通信制御方式

高野了成[†1]　工藤知宏[†1]　児玉祐悦[†2,†1]

仮想計算機の入出力性能を向上するため、仮想計算機モニタをバイパスするハードウェア支援機構が実用化されている。本研究では、SR-IOV デバイスを有する仮想計算機上のネットワーク QoS 制御を実現するために、仮想計算機と仮想計算機モニタが協調して動作するトラフィック監視および送信レート制御方式を提案する。提案方式のプロトタイプを KVM 上に実装し、期待通りの動作を確認した。

# Traffic monitoring and control method for virtual machine monitor-bypass network devices

Ryousei Takano,[†1] Tomohiro Kudoh[†1]
and Yuetsu Kodama[†1]

Virtual machine monitor-bypass device technologies have been turned to practical use in order to improve the I/O performance on virtual machines. This paper proposes traffic monitoring and transmission rate control method cooperated with a virtual machine monitor and virtual machines. It achieves precise and dynamic network QoS control even if virtual machines shares SR-IOV devices. We have implemented a prototype of the proposed method on the KVM, and confirm it works properly.

†1 産業技術総合研究所 情報技術研究部門
Information Technology Research Institute, National Institute of Advanced Industrial Science and Technology (AIST)
†2 筑波大学 計算科学研究センター
Center for Computational Sciences, University of Tsukuba

## 1. はじめに

クラウドコンピューティングは、計算資源を抽象化して運用する手段として近年その利用が拡大している。中でも、計算機のコモディティ化の進展を背景に、計算機ハードウェアそのものを仮想化する Infrastructure as a Service (IaaS) が実用的なサービスとして提供されるようになった。IaaS は既存の物理サーバをそのまま仮想化できる点で柔軟性が高く、また複数のサービスを一台のハードウェアに集約する際のセキュリティにも優れている。しかし、IO インテンシブ処理を仮想計算機上で実行するには、仮想化に伴うオーバヘッドにより十分な性能が得られない。この問題に対する解決策の 1 つとして、仮想計算機モニタ（Virtual Machine Monitor, VMM）をバイパスする IO 仮想化技術が普及し始めている。本論文では IO 仮想化技術の一つである Single Root IO Virtualization (SR-IOV)[1] ネットワークデバイスに焦点を当て、その問題と解決策について議論する。

SR-IOV デバイスを利用した場合、仮想計算機上のゲスト OS は物理デバイスに対して、VMM を経由することなく直接アクセスできる。類似技術である PCI パススルーとの違いは、1 台の物理計算機上で動作する複数の仮想計算機からそのデバイスを共有できる点にある。VMM バイパスは性能面で優位になる反面、資源管理の面で新たな問題が発生する。

問題を明らかにするために、ギガビットイーサネットで接続された 2 台の物理計算機上にそれぞれ 2 つの仮想計算機を動作させ、仮想計算機間で TCP、もしくは UDP を用いて通信した際の送信レートの変化を**図 1** に示す。実験環境の詳細は 4 節で示すが、物理計算機間の利用可能帯域は 800 Mbps に制限している。実験では TCP 通信の開始から約 10 秒後に、別の仮想計算機間で UDP 通信を開始した。その結果、図より明らかなように、利用可能帯域は UDP 通信でほぼ占有されてしまい、公平性が大きく崩れてしまっている。この原因は TCP の輻輳制御の影響により送信レートを抑制することにある。

仮想計算機間の通信の公平性問題に対して、従来の仮想計算機環境では、VMM レベルで帯域制御を行うことで対処できた。しかし、VMM バイパス環境では、VMM が通信に介入できないので、これとは別の解決策を考える必要がある。具体的な要求としては、VMM から、仮想計算機ごとの通信状況のモニタリング、および送信レートの制限が必要となる。

本論文では、SR-IOV デバイスを有する仮想計算機上で柔軟なネットワーク QoS 制御を実現するために、仮想計算機と仮想計算機モニタが協調して動作するトラフィック監視および制御方式を提案する。まず、SR-IOV デバイスの持つ、統計情報および帯域制御機能をホスト OS から利用可能にする。さらに、ハードウェアの持つ高精度な帯域制御の利用をゲス



© 2011 Information Processing Society of Japan

Vol.2011-ARC-195 No.13
Vol.2011-OS-117 No.13
2011/4/13

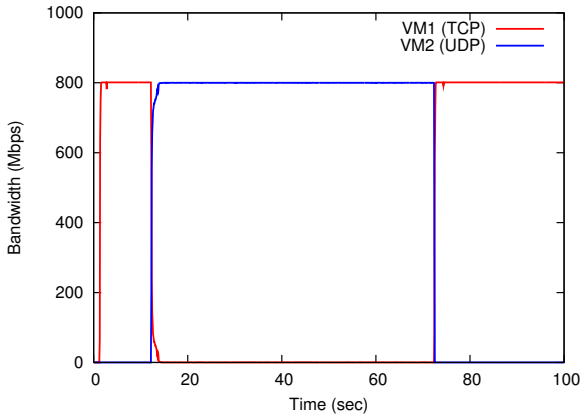


**図 1** 仮想計算機間通信における公平性 (送信レート制御なし)
Fig. 1 Fairness of inter-VM communication (without VM transmission rate control)

ト OS にも公開し、ゲスト OS とホスト OS の 2 段階での帯域制御を実現する。

## 2. 仮想計算機におけるデバイス制御

### 2.1 仮想化 IO 方式と直接 IO 方式

仮想計算機におけるデバイス制御を、デバイスの共有方式の違いの観点で比較したものを **図 2** に示す。なお、以降の説明ではネットワークデバイスを前提に説明する。

ソフトウェアベース共有方式（図 2 a）では、デバイスは仮想化され、ゲスト OS のゲストドライバは VMM が提供する仮想スイッチを介してパケットを授受する。物理デバイスを複数の仮想計算機から共有することに理論上の制限はなく、特殊なハードウェア支援も不要である。しかし、オーバヘッドが大きく性能上の問題が存在する。実装の詳細には違いがあるが、Xen のスプリットデバイスドライバや、KVM の virtio はこの方式に含まれる。

残りの 2 つは、VMM をバイパスして、ゲスト OS から物理デバイスに直接アクセスを可能にする方式である。直接割当て方式（図 2 b）は、1 つの仮想計算機に 1 つのデバイスを排他的に割り当てる。入出力性能は物理計算機に匹敵するが、CPU やチップセットが Intel VT-d などの仮想化支援機能に対応している必要である。一般的に PCI パススルーと呼ばれる技術はこの方式である。

SR-IOV 共有方式（図 2 c）は、直接割当て方式の共有制限を緩和するために PCI SIG

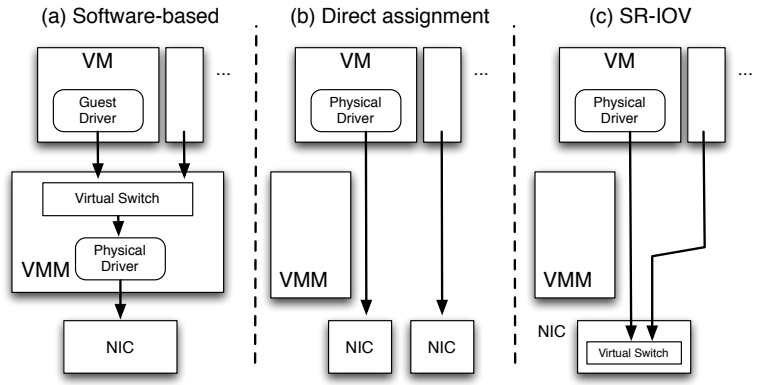


**図 2** 仮想化 IO モデル
Fig. 2 Virtualizaed IO models.

によって提案された方式である。SR-IOV デバイスは物理デバイスのすべての機能に対してアクセスできる Physical Function (PF) の他に、Virtual Function (VF) と呼ばれる仮想 PCI デバイスを提供し、仮想計算機ごとに VF を割り当てることができる。したがって、VF 数分の仮想計算機からデバイスを共有できる。VF 数は実装依存であるが、Alternative Routing-ID Interpretation (ARI) の制限上、256 個が上限と思われる。Intel 82576 は 8 個、82599 は 64 個の VF を有する。SR-IOV を利用するには、VT-d 以外にもデバイス側の対応が必要である。SR-IOV に対応したデバイスには、Intel 82576、82599 および Broadcom BCM57712 コントローラを搭載したイーサネット NIC、Mellanox ConnectX2 InfiniBand HCA などが存在するが、現在 Linux カーネルが対応しているものは Intel 製に限られている。Mellanox ConnectX2 の SR-IOV 対応ドライバに関しては、Mellanox で開発が進められている。

### 2.2 SR-IOV ドライバ

SR-IOV 共有方式におけるデバイスドライバの構成の概要について述べる。SR-IOV デバイスは、前述の通り PF と VF の 2 種類の機能を提供し、ホスト OS では PF ドライバが、ゲスト OS では VF ドライバが動作する。PF はフルセットの機能を提供する一方、VF の機能はそのサブセットとなり、不足する機能は PF に対して実行要求する必要がある。例えば、リンク状態が変更すると PF から VF へ通知が発生し、ゲスト OS が VLAN の設定を行うときは、VF から PF への通信が発生する。PF-VF 間におけるメッセージ送信やイベ



© 2011 Information Processing Society of Japan

Vol.2011-ARC-195 No.13
Vol.2011-OS-117 No.13
2011/4/13

ント通知に用いる通信チャネルに関して、SR-IOV では定義されていないが、Intel の実装ではメイルボックとドアベル機能を提供している。

本論文で対象とする Intel 製のギガビットイーサネットコントローラチップ 82576 [2] では、メイルボックス用に各 VF 毎に 64 バイトのメモリを NIC に内蔵しているので、この通信チャネルを用いて最大 64 バイトの任意のデータを授受できる。

## 3. 設計と実装

### 3.1 概　　要

本論文では、SR-IOV ネットワークデバイスを共有する仮想計算機環境において、ホスト OS からトラフィックをモニタリング、帯域制御する方式を提案する。設計の概要を **図 3** に示す。本方式のプロトタイプを Linux/KVM 環境上の SR-IOV ドライバを拡張することで実現する。

帯域制御として、VMM 側で各仮想計算機の送信レートのハードリミットを、仮想計算機ごとにソフトリミットを設定できるようにする。実際の送信レートはソフトリミットに従って制御され、ソフトリミットの値はハードリミットを越えることはできない。これは UNIX の資源使用制限である ulimit と同じ考え方である。送信レート制御は PF 側からのみ制御できるので、VF は PF に処理を委譲する。また、仮想計算機毎のモニタリング機能は PF ドライバが持つ。

以上のドライバ機能により、ホスト OS 上の管理アプリケーション（Traffic engineering controller）から、各仮想計算機の通信状況を把握することが可能になり、動的に送信レートへフィードバックを施すことが可能になる。

### 3.2 ホスト OS からの送信レート制限

Intel 82576 および 82599 コントローラチップは、PF ドライバにおいて、VF ごとの送信レートを制御可能である。これを VM 送信レート制御と呼ぶ。以下、82576 の場合について説明する。まず、関連レジスタセットを次に示す。

- VMBACS: VM Bandwidth Allocation Control & Status
- VMBAMMW: VM Bandwidth Allocation Max Memory Window
- VMBASEL: VM Bandwidth Allocation Select
- VMBAC: VM Bandwidth Allocation Config

制御の大まかな流れとして、VMBACS で機能を有効にした後、VMBASEL で制御対象の VF を選択し、VMBAC に目標送信レートを設定する。目標送信レート（target rate）は物理帯域（link speed）に占める割合であり、*link speed*/*target rate* から計算できる。制御の解像度は、整数部 10 ビット、小数部 14 ビットである。例えば、物理帯域が 1 Gbps の場合、目標送信レートを 500 Mbps に設定するには 0.5 に設定する。

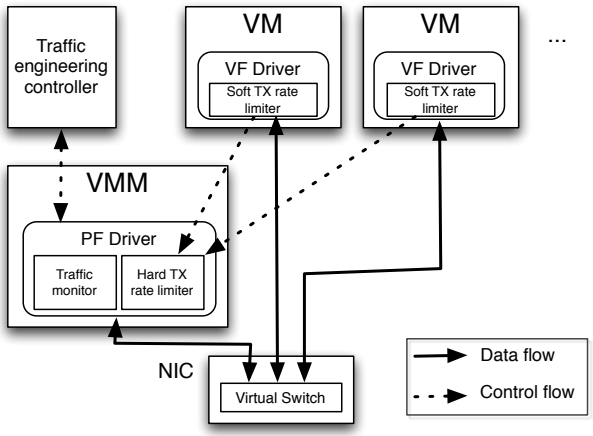


**図 3** 提案方式の概要
Fig. 3 Overview of the proposed method.

本機能は Linux 標準の igb デバイスドライバでは利用されていないが、Simon Horman により e1000-devel リストに投稿されたパッチ[3] を基にしている。ユーザインタフェースとして sysfs を利用し、`/sys/class/net/ethX/device/bandwidth_allocation` に VF ごとの目標送信レートを書き込むことでレジスタに反映する。例えば、最大 VF 数が 4 で、VF0 を 500 Mbps、VF2 を 200 Mbps に設定する場合は、次のように実行する。なお、目標送信レートが 0 は送信レートを制限しないことを意味する。

```
# echo /sys/class/net/eth0/device/bandwidth_allocation
# 0 0 0 0
# echo "500 0 200 0" > /sys/class/net/eth0/device/bandwidth_allocation
# echo /sys/class/net/eth0/device/bandwidth_allocation
# 500 0 200 0
```

VM 送信レート制御を用いて送信レートを 500 Mbps に設定し、そのパケットをキャプチャし、パケット送信間隔を測定した。その結果、パケット間ギャップ（Inter Packet Gap,



© 2011 Information Processing Society of Japan

IPG）は、MTU 1500 バイトで理想的な 72 マイクロ秒に制御されていることがわかった。また、NIC のオフロード機能の一つである TCP Segmentation Offload (TSO) を用いた場合、64 KB 分のパケットがバースト送信される[4]。しかし、VM 送信レート制御は TSO の後段で適用されるので、TSO 有効時でも IPG が正確に保たれていた。

さらに、2 つの仮想計算機のそれぞれの送信レートを 500 Mbps に制限した場合、両者のパケットが交互に送信されることを確認した。これらの観測結果から、VM 送信レート制御は、トークンバケットによる一般的なシェーピングではなく、パケットペーシングに極めて近い帯域制御を行っていると推測できる。ペーシングは帯域遅延積の大きなネットワーク等での性能改善効果が知られており[4]、これをゲスト OS から制御できることは有用と考える。

### 3.3 ゲスト OS からの送信レート制限

ゲスト OS における帯域制御としては、Linux 標準の `tc` コマンドが利用可能であるが、上記したハードウェアが提供する高精度なレート制御を使用したいという要求も考えられる。利用者は機能性と精度のトレードオフを考慮し、方式を選択すればよい。

使用例を次に示す。read 時の第 1 カラムはハードリミット、第 2 カラムはソフトリミットとなる。ゲスト OS から制御できるのはソフトリミットだけであり、またハードリミットを越える値は設定できない。

```
# echo /sys/class/net/eth0/device/tx_rate
# 500 0
# echo 400 > /sys/class/net/eth0/device/tx_rate
# echo /sys/class/net/eth0/device/tx_rate
# 500 400
```

### 3.4 モニタリング

Intel 82576 および 82599 コントローラチップが提供する VF ごとの統計情報関連レジスタを次に示す。VF は取得できる統計情報に関しても制限がある。

- VFGPRC: Good Packets Received Count
- VFGPTC: Good Packets Transmitted Count
- VFGORC: Good Octets Received Count
- VFGOTC: Good Octets Transmitted Count
- VFMPRC: Multicast Packets Received Count

82599 では VFGORC および VFGOTC フィールドのビット数が 36 ビットに拡張されているなどの違いはあるが、基本的に同じである。また、PF と異なりエラー回数を知ることはできない。

VF は自身の統計情報だけにアクセスすることが許されるが、PF は全 VF に対する統計情報にもアクセスできる。しかし、現在の Linux デバイスドライバには該当する処理が実装されていないので、sysfs インタフェース経由で上記の統計情報を取得できるように機能拡張した。

## 4. 実　　験

### 4.1 実験環境

実験には SR-IOV 対応ギガビットイーサネット (GbE) を有する 2 台の計算機 Dell PowerEdge T410 を用いた。各計算機は、Intel hexa-core Nehalem (X5650 2.66 GHz) 1 基、メモリ 6 GB などから構成される。イーサネットには Intel 82576 コントローラを搭載した 2 ポートギガビットイーサネット NIC E1G42ET を用いた。また、OS は Debian 6.0 を用いた。Linux カーネルのバージョンは 2.6.32-5-amd64 であり、KVM は本カーネルに含まれるモジュールを用いた。各仮想計算機には 1 vCPU とメモリ 1GB を割り当てた。

2 台の計算機はトラフィック観測用のハードウェアネットワークテストベッド GtrcNET-1[5] を介して接続した。今回の実験では、ボトルネックリンクの模擬や 100 ミリ秒間隔での帯域測定に GtrcNET-1 を用いた。

### 4.2 予備評価

提案方式の評価を行う前に、予備評価として非仮想化環境（Bare metal）と図 2 に示した 3 つの IO 仮想化方式の基本性能を比較した。測定項目として、TCP をバルク転送した際のグッドプット（アプリケーションレベルのスループット）と、1 秒間あたりに 16KB メッセージの ping pong を何回実行できるかというトランザクション数を選択した。ベンチマークには netperf を用い、3 回試行した最大値を結果とした。

結果を **表 1** に示す。これよりグッドプットには違いは見られないものの、トランザクション数は Bare metal、Direct attached、SR-IOV の順で大きく低下している。これは仮想計算機モニタによる割込みインジェクションのオーバヘッドにより、通信遅延が増加していることが原因と考える。Intel VT-d を用いることで、ゲスト OS の仮想アドレスから実アドレスへの変換や、MSI/MSI-X 割込みのリマップをハードウェアにオフロードできるが、VMM は仮想計算機に対して割込み発生を通知する必要があり、オーバヘッドとなり得る[6]。この処理を割込みインジェクションと呼ぶ。


© 2011 Information Processing Society of Japan

Vol.2011-ARC-195 No.13
Vol.2011-OS-117 No.13
2011/4/13

**表 1** netperf ベンチマークによるグッドプットとトランザクション数
Table 1 Goodput and transaction per second using the netperf benchmark.

| | goodput | trans. per second |
|---|---|---|
| Bare metal | 941.26 | 19108.88 |
| Direct attached | 941.45 | 8497.49 |
| SR-IOV | 941.45 | 2049.08 |

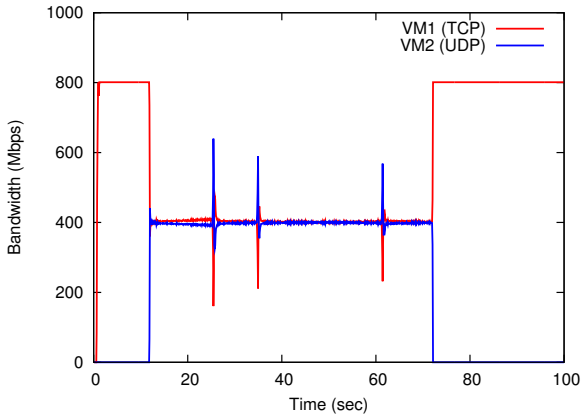


**図 4** 仮想計算機間通信における公平性 (送信レート制御あり)
Fig. 4 Fairness of inter-VM communication (with VM transmission rate control)

### 4.3 送信レート制限

ホスト OS からの送信レートのハードリミット制御の効果を確認するため、論文の冒頭で示した 図 1 と同様の実験を行った。2 台の物理計算機間の利用可能帯域は 800 Mbps である。提案方式を用いて、UDP 通信の送信側の仮想計算機の送信レートを 400 Mbps に制限した。結果を **図 4** に示す。

UDP 通信の送信レートは 400 Mbps に制限されたので、TCP 通信の送信レートも 400 Mbps を維持しており、仮想計算機間の通信は公平になっている。ただし、3 箇所で UDP 通信のスパイクが観測された。原因の追及は今後の課題であるが、NIC の送信レート制御がうまく働かず、UDP パケットがバースト的に送信されたものと考える。

また、ゲスト OS からの送信レートのソフトリミット制御も正常に動作することも確認した。

## 5. 議　　論

SR-IOV のようなデバイスモデルは、従来の OS では想定外であったので、現時点で確立したインタフェースは存在しない。Linux では通常のネットワークデバイスは eth0 などのデバイス名が割り当てられ、`ifconfig` や `ethtool` 等のコマンドから、ioctl や netlink のインタフェースを介して制御される。PCI パススルーを用いる場合、ホスト OS 側では、該当デバイスに対してスタブドライバを割当てることで、ネットワークデバイスとしての機能の使用を強制的に禁止する。したがって、ゲスト OS に PCI パススルーしたデバイスをホスト OS のユーザランドから見えないので、ホスト OS からゲスト OS を監視・制御することができなかった。

提案方式の現在の実装は、PF ドライバに付属する情報として sysfs インタフェースを採用しているが、利便性を考えると、より一般的なコマンドとの親和性の高いインタフェースを考える必要がある。例えば、VF を veth0.1 のような疑似デバイスとして見せる方法が考えられる。ピリオド後の数字は VF 番号である。次にコマンドからの使用例を示す。

```
# ethtool --statistics veth0.1        (統計情報の表示)
# ethtool --tx-rate veth0.1           (最大送信レートの取得)
# ethtool --tx-rate veth0.1 500mbit   (最大送信レートの設定)
```

これを実現するためには netlink インタフェースを有する VF ネットワークデバイス用のスタブドライバを実装するなどの方法が考えられる。

SR-IOV 対応 NIC は、仮想計算機（すなわち VF）間のネットワークをブリッジするために、スイッチを内蔵している。これは Virtual Ethernet Bridge (VEB) と呼ばれる。送信レート制限はそのスイッチへの入力の手前で実施されるので、サーバ間通信だけではなく、同一サーバ上の仮想計算機間通信もその影響を受けることになる。この対策として、仮想計算機間の通信用に準仮想化ドライバを併用する解決策が考え得るが、ゲスト OS では、複数ネットワークが共存するマルチホーミング状態になるので、運用が難しくなる等の問題が残る。なお、NIC 内ではなく、イーサネットフレームに VM を識別するタグを付加することで、外部のスイッチにより VEB を実現する VEPA や VN-Tag などの規格も標準化されつつあるが、これらに関しても同様な問題があると考える。

## 6. 関 連 研 究

仮想計算機上の通信性能を改善する試みとして、(1) 仮想化デバイス処理の最適化、(2)



© 2011 Information Processing Society of Japan

Vol.2011-ARC-195 No.13
Vol.2011-OS-117 No.13
2011/4/13

通信を考慮した仮想計算機スケジューリングの改善、(3) VMM バイパス技術の利用が考えられる。前者 2 つは主に準仮想化による性能改善であり、本論文では (3) に焦点を合わせ、関連する問題について取り上げた。

仮想環境において、InfiniBand や 10 GbE などの高速インターコネクトの性能を最大限引き出すために、準仮想化ドライバの考えを応用して VMM をバイパスする機構が提案されている[7]。この機構は VT-d などハードウェアの支援を必要としないが、VMM ごとに準仮想化ドライバを実装する必要があり、現時点では Xen 以外で利用できない。一方、CPU 命令セットや周辺デバイスなどハードウェア面でも仮想化への対応が進められており、PCI パススルーや SR-IOV による VMM バイパス機構が利用可能になっている。

近年、クラウドコンピューティングに対する高性能計算 (High Performance Computing, HPC) の高い需要を受け、Amazon EC2[8] の Cluster Compute Instance、Cluster GPU Instance など、HPC を意識した IaaS が現れ始めている。このような背景の中、VMM バイパス機構は、仮想化によるユーザの利便性と性能の追求の両立を目指すための重要な要素技術であると考える。HPC システムに特化した軽量 VMM である Palacios では、仮想化オーバヘッドを避けるため VMM バイパスを採用している。J. Lange ら[9] は、ゲスト OS の Linux から InfiniBand をパススルーで用いた場合、非仮想化環境と比較して性能劣化は 3%以内であると報告している。また、通信遅延を削減するために、割込み駆動ではなくポーリングベースとすることで非仮想化環境に匹敵できると述べている。

提案手法のプロトタイプ実装では、VMM・ゲスト OS 間通信に SR-IOV デバイスのメイルボックスを用いたが、この手法はハードウェア依存であり、Intel 以外のデバイスで利用可能とは限らない。ソフトウェアによる VMM・ゲスト OS 間通信方式として、J.Lange ら[10] は SymSpy、SymCall アップコールや H. Eiraku ら[11] の Socket outsourcing などが提案している。

## 7. まとめと今後の予定

本論文では、SR-IOV ネットワークデバイスを共有する仮想計算機環境において、仮想計算機ごとの送信レートをホスト OS とゲスト OS のそれぞれから制御し、さらにホスト OS からトラフィックを監視する方式を提案した。提案方式のプロトタイプを KVM 上に実装し、期待通りの動作を確認した。提案方式を利用することで、VMM とゲスト OSM が協調して動作するネットワーク QoS 制御を実現できると考える。本論文では GbE を用いてプロトタイプしたが、さらに 10 GbE や InfiniBand への対応も進めていきたい。

我々は HPC 向け IaaS の構築を構想しており、HPC アプリケーションの性能測定を通じて、PCI パススルーを経由した InfiniBand 利用の効果を検証している[12),13)]。今後は、本論文での成果を踏まえて、ユーザランドとのインタフェースを見直し、本機能を Open Stack 等のクラウドスタックに組み込むことで、より簡便な利用を実現したい。

## 参 考 文 献

1) PCI SIG: I/O Virtualization, `http://www.pcisig.com/specifications/iov/`.
2) Intel: Intel 82576 Gigabit Ethernet Controller Datasheet, `http://download.intel.com/design/network/datashts/82576_Datasheet.pdf` (2010).
3) Horman, S.: [E1000-devel] [rfc 0/3 v3] igb: bandwidth allocation, `http://www.mail-archive.com/e1000-devel@lists.sourceforge.net/msg02077.html`.
4) Takano, R., Kudoh, T., Kodama, Y. and Okazaki, F.: High-resolution Timer-based Packet Pacing Mechanism on the Linux Operating System, *Internet Conference 2010* (2010).
5) Kodama, Y., Kudoh, T., Takano, R., , Sato, H., Tatebe, O. and Sekiguchi, S.: GNET-1: Gigabit Ethernet Network Testbed, *Cluster 2004*, IEEE (2004).
6) 渡邉和樹，永島　力，茂田井寛隆，片山吉章，毛利公一：Xen における PCI Passthrough の性能評価，情報処理学会研究報告，Vol.2010-OS-113, No.3, pp.1–8 (2010).
7) Liu, J., Huang, W., Abali, B. and Panda, D.K.: High Performance VMM-Bypass I/O in Virtual Machines, *Proceedings of the USENIX Annual Technical Conference*, pp.29–42 (2006).
8) Amazon Elastic Compute Cloud (Amazon EC2): `http://aws.amazon.com/ec2/`.
9) Lange, J., Pedretti, K., Dinda, P., Bridges, P., Bae, C., Soltero, P. and Merritt, A.: Minimal-overhead Virtualization of a Large Scale Supercomputer, *The 2011 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE 2011)*, pp.169–180 (2011).
10) Lange, J. and Dinda, P.: SymCall: Symbiotic Virtualization Through VMM-to-Guest Upcalls, *The 2011 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE 2011)*, pp.193–204 (2011).
11) Eiraku, H., Shinjo, Y., Pu, C., Koh, Y. and Kato, K.: Fast Networking with Socket-Outsourcing in Hosted Virtual Machine Environments, *24th ACM Symposium on Applied Computing*, pp.310–317 (2009).
12) 池上　努，高野了成，田中良夫，中田秀基，関口智嗣：クラウドコンピューティングの性能評価，情報処理学会研究報告，Vol.2010-HPC-128, No.14, pp.1–6 (2010).
13) 高野了成，池上　努，広渕崇宏，田中良夫：InfiniBand を PCI パススルーで用いる仮想化 HPC クラスタの性能評価，先進的計算基盤システムシンポジウム（SACSIS 2011）(2011 採録予定).



© 2011 Information Processing Society of Japan