Vol.2019-OS-147 No.2
2019/7/24

ソフトウェアExpEtherの DPDKを用いた広帯域化に関する検討

鈴木 順[1,a)]　林 佑樹[1]

概要：コンピュータシステムの柔軟な構成や運用を実現するには，その用途や刻々と変化するサービス要求に応じてシステムの構成を柔軟に変更できるアーキテクチャを実現する必要がある．我々はI/O(Input/Output) デバイスに着目し，コンピュータのCPU(Central Processing Unit) とI/O デバイスを分離してEthernet で接続し，I/O デバイスを必要に応じてCPU を収容するホストに割り当てるExpEtherを提案した [1]．さらに，ExpEther はホストとI/O デバイスの双方にI/O バスであるPCI Express(PCIe) とEthernet をブリッジするハードウェアブリッジを必要としたため，ホスト側は仮想マシン (VM, Virtal Machine) のI/O 仮想化機構を拡張し，従来のNIC(Network Interface Card) を用いてEthernet 上のI/O デバイスと接続するソフトウェアExpEtherを提案した [2]．しかし，ソフトウェアExpEtherではPCIeのデータ長が短いパケットを生成，Ethernet に送出する処理や，I/O デバイスから受信したパケットを解析する処理をパケット毎にソフトウェアで行うため，Ethernet で接続したI/O デバイスのI/O 性能が制限されるという課題があった．そこで本稿では，このI/O 性能の課題を解決するため，新たにDPDK(Data Plane Development Kit) を用いたソフトウェアExpEtherのアーキテクチャを提案する．提案手法では，ソフトウェアExpEtherのプログラムがユーザ空間内でPCIeパケットの処理を行い，OS(Operating System) を介さずに直接NIC(Network Interface Card) を用いてPCIeパケットをカプセル化したEthernet フレームの送受信を行う．これにより，これまでI/O パケットの送受信毎に発生していた割り込みやカーネルモードへの遷移のオーバヘッドを削減する．提案手法を実装して評価を行った結果，[2] で提案した従来のソフトウェアExpEtherの構成よりI/O 帯域が最大で298%向上した．

Bandwidth Enhancement of Software ExpEther Using DPDK

Jun Suzuki[1,a)]　Yuki Hayashi[1]

1. はじめに

コンピュータシステムではその用途や刻々と変化するサービス要求に応じてシステムの構成を柔軟に変更できるアーキテクチャが望まれる．多くのWebシステムや分散ミドルウェアでは，この要求をホスト数の増減で対応するスケールアウト型のアーキテクチャによって実現している [3]．一方，各ホストの内部に目を向けると，それぞれのホストはCPU(Central Processing Unit)，ブリッジ，メモリ，I/O デバイス等で構成されており，これらのデバイスも必要に応じて柔軟に組み合わせてサービスを提供できることが望ましい．そのようなアーキテクチャを実現することにより，データセンターだけでなく，組み込みや社会インフラ等の幅広いコンピュータシステムの実装制約や熱問題を解消できたり，サービス要求に応じてデバイスの粒度でシステムの処理量を調整することが可能になる．

著者らは特にI/O(Input/Output) デバイスの柔軟な構成に着目し，複数のホストとI/O デバイスをEthernet を用いて接続し，I/O デバイスを必要に応じて必要なホストに割り当てるExpEther技術を提案してきた [1], [2]．また他所でもこのようなI/O デバイスの柔軟な構成を可能とする技術には，ハードウェアでは独自のインターコネクション

[1] NEC デジタルプラットフォーム事業部
1753, Shimonumabe, Nakahara-Ku, Kawasaki, Kanagawa 211–8666, Japan
[a)] j-suzuki@ax.jp.nec.com

を用いる手法 [4] や，PCIe を拡張して複数のホストを接続し，I/O デバイスを保持するホストに I/O 処理を依頼する手法 [5] が提案されている．また産業界からはホストの I/O バスを拡張する Thunderbolt の規格の普及が進められている [6]．一方ソフトウェアでは，USB(Universal Serial Bus) を IP(Internet Protocol) で拡張する手法 [7] や，SCSI(Small Computer System Interface) の通信を TCP/IP を用いて送受信する iSCSI が提案されている [8]．

ExpEther [1] はホストと I/O デバイスを Ethernet を用いて接続し，ホストと I/O デバイスの間で PCI Express(PCIe) の I/O パケットを Ethernet フレームにカプセル化し通信を行う手法である．ホスト及び I/O デバイスの Ethernet への接続にはハードウェアの ExpEther ブリッジを用いる．Ethernet で接続した I/O デバイスは，任意のホストに割り当てることができる．ExpEther では，I/O デバイスが PCIe に準拠していればあらゆるデバイスを接続することができ，OS(Operating System) や I/O デバイスのドライバを変更する必要がない．これにより，ExpEther を用いることで，様々な I/O デバイスを任意のホストに柔軟に割り当てる汎用インターコネクションが実現される．

また ExpEther では，ホストと I/O デバイスの双方に PCIe と Ethernet をブリッジするハードウェアの ExpEther ブリッジが必要だったため，著者らはホスト側の ExpEther ブリッジを不要とするソフトウェア ExpEther を提案した [2]．ソフトウェア ExpEther は，仮想マシン (VM, Virtual Machine) の I/O 仮想化機構を拡張し，VM の I/O トランザクションをソフトウェアで監視する．そして，取得した I/O 命令から，従来ハードウェアが作成していた PCIe の I/O パケットをソフトウェアで作成し，ホストの NIC(Network Interface Card) から Ethernet で接続した I/O デバイスに送出する．これにより，ホスト側はハードウェアの ExpEther ブリッジを使用することなく，従来の NIC を介して Ethernet で接続した I/O デバイスを使用可能となる．

ここでソフトウェア ExpEther は，I/O パケットのソフトウェア処理のオーバヘッドのため，Ethernet で接続した I/O デバイスの I/O 性能が制限されるという課題があった．一般に PCIe の I/O パケットは，データ長が 128B や 256B と短いパケットが多い．そのため I/O 命令の解析や PCIe パケットの作成，Ethernet フレームへのカプセル化と送受信をユーザ空間で行うソフトウェア ExpEther では，頻繁にカーネルモードへの遷移や，パケット受信による割り込み処理が発生する．このオーバヘッドのため，ソフトウェア ExpEther の I/O パケットに関するソフトウェア処理が I/O 性能のボトルネックとなっていた．

本稿では，このようなソフトウェア ExpEther の性能課題を解決するため，DPDK(Data Plane Development Kit) [9] をソフトウェア ExpEther に適用したアーキテクチャを提案する．DPDK では NIC を利用するアプリケーションプログラムが OS を介さずポーリングで直接 NIC を制御することが可能となる．これにより，パケット処理に伴うカーネルモードへの遷移や，パケット受信による割り込みのオーバヘッドを削減できる．提案手法では，ソフトウェア ExpEther を VM のホスト OS のアプリケーションプログラムとして実装し，I/O パケットの生成と Ethernet フレームへのカプセル化をソフトウェア ExpEther 内で実施して，カーネルを介さずに NIC から Ethernet で接続した I/O デバイスに送受信を行う．

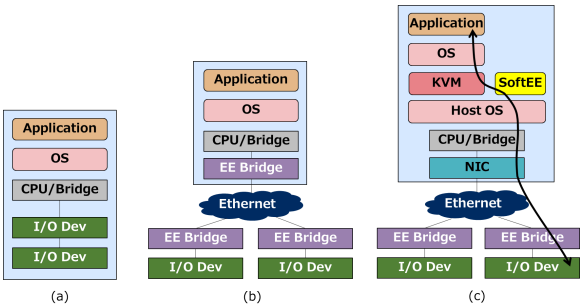


図 1　I/O デバイス拡張方式の比較．(a) 従来の構成，(b)ExpEther，(c) ソフトウェア ExpEther．EE: ExpEther

提案手法を KVM(Kernel-based Virtual Machine) に実装し評価を行った結果，従来のソフトウェア ExpEther の構成に対し最大で 298%の帯域向上効果が得られた．

以下本論では第 2 節で提案の背景として ExpEther, ソフトウェア ExpEther, DPDK を述べ，続いて第 3 節で DPDK を用いたソフトウェア ExpEther のアーキテクチャを提案し，第 4 節で性能評価について述べ，最後に第 5 節でまとめる．

2. 背景

2.1 ExpEther

ExpEther ではホストと I/O デバイスを Ethernet を用いて接続する．図 1(b) に ExpEther の構成を示す．ホスト側の ExpEther ブリッジは HBA(Host Bus Adaptor) として実装され，ホストの I/O スロットに挿入される．これにより，ホストの CPU ブリッジと ExpEther ブリッジが PCIe バスで接続される．また HBA は Ethernet インターフェースを保持しており，ホストを Ethernet に接続する．一方 I/O デバイス側の ExpEther ブリッジは，PCIe に準拠した I/O デバイスを複数収容する I/O Box の基盤に実装される．この I/O Box に収容された I/O デバイスは，I/O Box の Ethernet インタフェースを通して任意のホストに割り当てることができる．図 2 に ExpEther の HBA と I/O Box の実装の一例を示す．

ExpEther で接続されたホストと I/O デバイスの間では，PCIe の I/O パケットが Ethernet フレームにカプセル化され送受信される．PCIe ではホストと I/O デバイスの間の

(a) ExpEther HBA.

(b) ExpEther I/O Box.
図 2 ExpEther の HBA と I/O Box.

通信がパケットを用いて実現されており，その I/O パケットを Ethernet フレームにカプセル化する．このパケット処理はブリッジのハードウェア機構で行われるため，処理速度は従来ホスト内で用いられてきた PCIe バスのブリッジと同程度の高速性が実現される．また，Ethernet においては複数のホストや I/O デバイスの間で通信が同時に発生し，輻輳により通信遅延が増大することを防ぐため，ExpEther ブリッジの間で通信遅延に基づく送信レート制御を行っている．ExpEther はまた，ExpEther ブリッジ間で Go-Back-N ARQ(Automatic Repeat Request) による欠落パケットの再送を行っており，ExpEther ブリッジ間で高信頼な I/O パケットの伝送を実現している．

ExpEther では，Ethernet の領域はホストに対し単一の分散 PCIe スイッチとして仮想化される．PCIe スイッチとは PCIe において I/O バスのファンアウト機能を提供するデバイスである．そのため，Ethernet はホストに対して透過となり，ExpEther で接続した I/O デバイスはホストの I/O デバイスのツリーの配下に現れる．またこれらのデバイスは従来の OS やデバイスドライバを変更せずに用いることができる．また I/O デバイスの接続に利用する Ethernet スイッチも標準品である．

2.2 ソフトウェア ExpEther

ExpEther ではホスト側と I/O デバイス側の双方にハードウェアの ExpEther ブリッジが必要だった．ハードウェアの ExpEther ブリッジは広帯域低遅延の接続を提供する一方，専用のハードウェアを必要とするため ExpEther をシステムに導入する際の制限となっていた．例えばタブレット等のコンシューマ機器や組み込み向けの小型ホストでは，ExpEther HBA を挿入する I/O スロットを実装していなかったり，空きスロットがない場合があった．

そこでソフトウェア ExpEther は，ホストを VM システムとし，VM の I/O 仮想化機構を ExpEther 向けに拡張することでホスト側の ExpEther ブリッジを不要とした．これにより，ホストを NIC 等の標準のネットワークインタフェースを介して Ethernet 上の I/O デバイスと接続し，I/O デバイスを利用可能とした．

ソフトウェア ExpEther の構成を図 1(c) に示す．ソフトウェア ExpEther は，KVM のパススルーモードを拡張して実装されている．拡張した KVM では VM の I/O トラフィックを監視する．そして，I/O トラフィックが Ethernet で接続した I/O デバイスに対する命令であった場合，命令をホスト OS のユーザプログラムとして動作するソフトウェア ExpEther に渡す．ソフトウェア ExpEther では I/O 命令の内容を解析し，従来 CPU ブリッジがハードウェアで作成していた I/O パケットを作成し，Ethernet フレームにカプセル化し，NIC から Ethernet 上の I/O デバイスに送信する．ソフトウェア ExpEther はまた，イーサネットで接続した I/O デバイスから受信した DMA(Direct Memory Access) 要求の処理も行う．

ソフトウェア ExpEther は KVM の I/O 仮想化機構を拡張して実装されるため，VM 内の OS やデバイスドライバの変更の必要がない．また KVM のホスト OS も変更不要である．またソフトウェア ExpEther の実装は，最大限ホスト OS のアプリケーションプログラムとして実現されており，KVM 自体の拡張は最小に留めている．

2.3 DPDK

DPDK は Linux Foundation の下で開発が進められているソフトウェアフレームワークである．DPDK を用いることにより，ソフトウェアで高速なネットワークアプライアンスを実現することが可能となる．DPDK では，ユーザプログラムとして実装されたネットワークアプリケーションが OS を介さず直接 NIC を制御する．この目的のため，DPDK は NIC をアプリケーションプログラムから直接利用するための PMD(Poll Mode Driver) とネットワークプログラムを作成するための関数を提供するライブラリから構成される．

DPDK で NIC をポーリングで利用するアプリケーションプログラムは，ホストの CPU コアを占有し，CPU 使用率 100%で動作する．それによりアプリケーションプログラムからネットワークに送受信を行うためのカーネルモードへの遷移やパケットのデータコピーのオーバヘッドを削

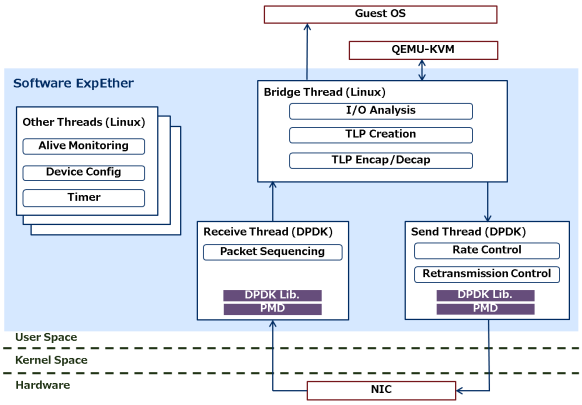


図 3 DPDK を用いたソフトウェア ExpEther のアーキテクチャ．

減する．これにより DPDK を用いることで広帯域，低遅延のネットワークアプライアンスを実現することが可能となる．

3. DPDK を用いたソフトウェア ExpEther

本稿で提案する DPDK を用いたソフトウェア ExpEther のアーキテクチャを図 3 に示す．ソフトウェア ExpEther は KVM のホスト OS のアプリケーションプログラムとして動作する．VM のデバイスドライバが発行した I/O 命令は，拡張した QEMU-KVM により検出されソフトウェア ExpEther に渡される．ソフトウェア ExpEther は渡された命令を基に Ethernet で接続した I/O デバイスに I/O パケットを発行する．ソフトウェア ExpEther はまた，Ethernet で接続した I/O デバイスから I/O パケットを受信する．このときソフトウェア ExpEther は，受信した I/O パケットの内容を基に，ゲスト OS のデバイスドライバから指定されたメモリ領域に直接 Read/Write を行う DMA 処理を実行するか，以前にデバイスドライバが I/O デバイスに送信した I/O 命令に対する応答を，QEMU-KVM を介して VM に返す．

ソフトウェア ExpEther の内部はブリッジスレッド，送信スレッド，受信スレッド，その他のスレッド群から構成される．

ブリッジスレッドは DPDK ではなく通常の Linux スレッドとして実装され，QEMU-KVM から渡された I/O 命令の解析，解析結果に基づく I/O パケットの作成，I/O パケットの Ethernet フレームへのカプセル化を行う．また Ethernet で接続した I/O デバイスから受信した Ethernet フレームのデカプセル化処理も行う．PCIe では I/O パケットを TLP(Transaction Layer Packet) と呼ぶ．

送信スレッドは DPDK スレッドとして実装され，ブリッジスレッドから渡された TLP がカプセル化された Ethernet フレームを OS を介さずに直接 NIC から I/O デバイスに送信する．このとき送信スレッドは Ethernet での輻輳を抑制する送信レート制御と，欠落したパケットを再送する処理を行う．送信レート制御及び再送制御は，これまで ExpEther で採用してきた方式と同じである．

受信スレッドも DPDK スレッドとして実装される．受信スレッドは Ethernet から受信したフレームの順序保障を実施し，受信したフレームをブリッジスレッドに渡す．また受信スレッドは受信フレームに対する Ack フレームを作成し，送信スレッドに Ack フレームの送信を依頼する．

その他のスレッド群では死活監視機能，デバイスコンフィギュレーション機能，タイマ機能を実装している．これらの機能は全て Linux スレッドによる実装である．死活監視機能はホストに割り当てられた I/O デバイスが接続するハードウェアの ExpEther ブリッジの死活監視を行っている．この死活監視のため，ソフトウェア ExpEther の死活監視機能とハードウェアの ExpEther ブリッジは定期的に死活監視パケットを Ethernet にブロードキャストする．死活監視機能は I/O デバイスが接続する ExpEther ブリッジから死活監視パケットを受信することにより，ホストに割り当てられた I/O デバイスが接続する ExpEther ブリッジの MAC(Media Access Control) アドレスを認識し，ブリッジスレッドが TLP をカプセル化する際の MAC アドレスとして使用する．

デバイスコンフィギュレーション機能は，Ethernet を介して接続した I/O デバイスを VM が利用する前にコンフィギュレーションを行う．この処理は I/O デバイスへのメモリマップアドレスや割り込みアドレスの割り当てを含む．これらのコンフィギュレーションに用いた情報は，ブリッジスレッドが QEMU-KVM から渡された I/O 命令から TLP を作成する際のヘッダ情報として使用される．

タイマ機能は I/O デバイスへの Ethernet フレームの伝送時に欠落したフレームの再送や，受信した Ethernet フレームに対する Ack フレームの送信のトリガに使用される．

なお死活監視パケットは，同じグループ番号でグルーピングされた ExpEther のノードだけで受信することができる．グループ番号はソフトウェア ExpEther のホストやハードウェア ExpEther ブリッジ毎に割り当てられ，任意の番号に変更可能である．これにより，1 台のホストと複数の I/O デバイスがグルーピングされ，I/O デバイスが同じグループ番号を保持するホストに割り当てられる．I/O デバイスを割り当てるホストを変更する場合，I/O デバイスが接続するハードウェア ExpEther ブリッジのグループ番号を変更する．

4. 性能評価

本節では KVM に DPDK を用いたソフトウェア ExpEther を実装し評価を行った結果を述べる．評価に用いた実験系を図 4 に，評価機材の一覧を表 1 に示す．評価では DPDK 適用前後のソフトウェア ExpEther，ハードウェア ExpEther，ベースラインとして I/O デバイスをホストの

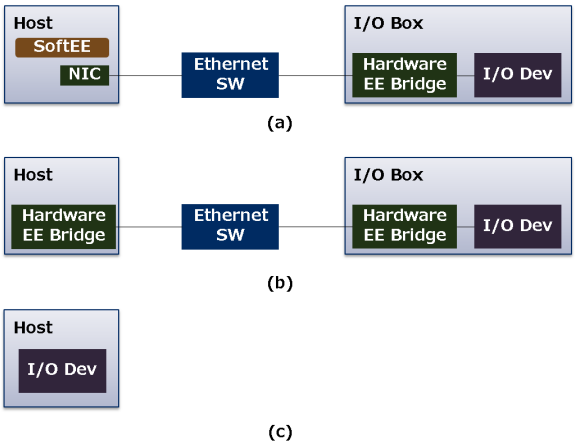


図 4　評価構成．(a) ソフトウェア ExpEther．(b) ハードウェア ExpEther．(c) 従来構成

表 1　評価で用いた機材

Host		NEC Express 5800 53Xh
CPU		Intel® Xeon® CPU E3-1275 v3
Memory		32GB
Host OS		CentOS 7.3
VM	Memory	2GB
	OS	CentOS 7.3
Ethernet Switch		NEC QX-S708B
Ethernet		1GbE
I/O Device	NIC	Intel® Gigabit ET Dual Port Server Adapter 82576
	SATA Controller	Silicon Image Inc Sil3132 Serial ATA RAID II Controller

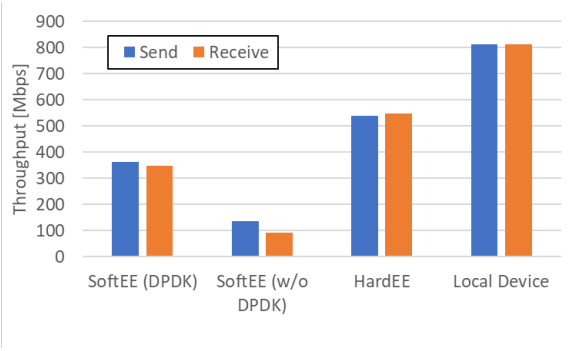
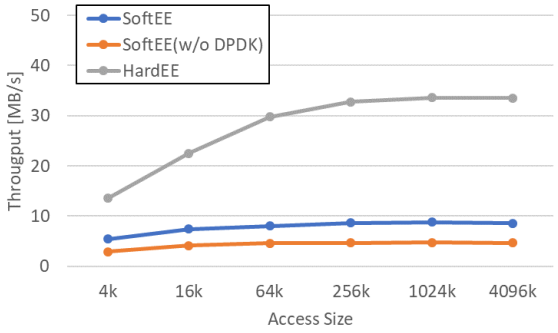
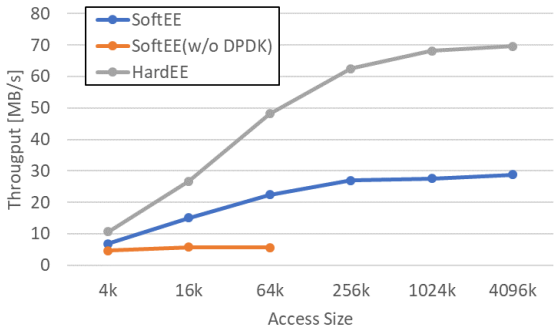


図 5　iperf による NIC 性能評価．

I/O スロットに直接挿入した場合の性能を比較した．ソフトウェア ExpEther の評価では，I/O デバイスの Ethernet の接続にはハードウェア ExpEther ブリッジを用いた．またホストと I/O デバイスは 1 台の Ethernet スイッチを用いて接続した．接続に用いた Ethernet は 1GbE である．また評価用の I/O デバイスとして，NIC 及び SATA(Serial ATA) コントローラを用いた．SATA コントローラと接続するストレージデバイスには，Intel X25-E SSD を使用した．NIC の性能評価ツールには iperf，SATA には fio を用いた．NIC の評価時には，NIC の通信相手として対向ホストを接続した．



(a) Write



(b) Read

図 6　fio による SSD 性能評価．

図 5 に iperf を用いた NIC の性能評価結果を示す．評価は UDP 通信で行い，通信レートを 1Gbps に設定した．ソフトウェア ExpEther を DPDK を用いて実現したことにより，DPDK を用いなかった従来の構成と比較して送信帯域が 165%，受信帯域が 279%向上した．DPDK を用いたソフトウェア ExpEther の性能は，I/O スロットに I/O デバイスを直接挿入した場合や，ハードウェア ExpEther ブリッジを使用した場合の性能には及ばない．しかしソフトウェア ExpEther では，特殊なハードウェアを必要とせず，ソフトウェアで Ethernet で接続した I/O デバイスが利用可能となるため，コンシューマや組み込み向けのデバイス，あるいはデータセンターにおける AI(Artificial Intelligence) 向けのアクセラレータで計算インテンシブな処理等への幅広い適用が期待される．

次に図 6 に fio を用いた SATA コントローラの性能評価結果を示す．評価は Write, Read 共ランダムアクセスとし，blocksize を変更しながら行った．並列 I/O は行わず，fio の numjobs は 1 に設定した．なお，SATA コントローラをホストの I/O スロットに挿入する評価は VM に SATA コントローラをパススルーモードで接続できなかったため実施しなかった．また，DPDK を用いないソフトウェア ExpEther の Read 性能評価は，256KB 以上のブロックアクセスでエラーが発生したため測定を行わなかった．原因を解析中だが，I/O 負荷が増大し，ソフトウェア ExpEther の処理が追い付かなくなったためだと考えている．

SATA コントローラを用いた評価でも，NIC を用いた評価と同様に，ソフトウェア ExpEther に DPDK を適用することにより，DPDK を用いない従来の構成と比較して最大で 298%の帯域向上が得られた．

5. まとめ

本稿では VM の I/O 仮想化機構を拡張し，ホストの NIC を介して Ethernet で接続した I/O デバイスの使用を可能とするソフトウェア ExpEther において，DPDK を適用して高い I/O 性能を実現するアーキテクチャを提案した．提案手法では，DPDK の適用により，従来のアーキテクチャにおいて短いデータ長の I/O パケットの処理毎に発生していた割り込み処理やカーネルモードへの遷移に関するオーバヘッドを削減した．これらのオーバヘッドはソフトウェア Expether の性能ボトルネックだった．提案手法を KVM に実装し NIC と SATA コントローラを用いて評価を行った結果，最大で 298%の性能向上効果が得られた．ソフトウェア ExpEther は従来のアプリケーションや OS を変更せず，空き I/O スロットがないホストに対してもネットワークで I/O デバイスの拡張を実現するため，データセンターだけでなく組み込みやコンシューマ機器等の幅広い用途への応用が期待される．

謝辞 本成果（の一部）は，内閣府が進める「戦略的イノベーション創造プログラム（SIP）第 2 期／フィジカル空間デジタルデータ処理基盤」（管理法人：NEDO）における研究開発によるものです．

参考文献

[1] Suzuki, J., Hidaka, Y., Higuchi, J., Hayashi, Y., Kan, M. and Yoshikawa, T.: Disaggregation and Sharing of I/O Devices in Cloud Data Centers., *IEEE Trans. Computers*, Vol. 65, No. 10, pp. 3013–3026 (2016).

[2] Suzuki, J., Tsuji, A., Hayashi, Y., Kan, M. and Miyakawa, S.: Device-Level IoT with Virtual I/O Device Interconnection, *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 67–74 (2016).

[3] Dean, J. and Ghemawat, S.: MapReduce: simplified data processing on large clusters, *Communications of the ACM*, Vol. 51, No. 1, pp. 107–113 (2008).

[4] Krishnan, V.: Evaluation of an Integrated PCI Express IO Expansion and Clustering Fabric, *Proc. 16th IEEE Symposium on High Performance Interconnects*, Stanford, CA, pp. 93–100 (2008).

[5] Hou, R., Jiang, T., Zhang, L., Qi, P., Dong, J., Wang, H., Gu, X. and Zhang, S.: Cost Effective Data Center Servers, *Proc. 2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA2013)*, Shenzhen, pp. 179–187 (2013).

[6] Intel: Thunderbolt 3, Intel (online), available from ⟨https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/thunderbolt-overview-brief.pdf⟩ (accessed 2019-06-24).

[7] Hirofuchi, T., Kawai, E., Fujikawa, K. and Sunahara, H.: USB/IP-A peripheral bus extension for device sharing over IP network, *Proceedings of the annual conference on USENIX Annual Technical Conference*, pp. 42–42 (2005).

[8] : Internet Small Computer Systems Interface (iSCSI), The Internet Society (online), available from ⟨https://tools.ietf.org/html/rfc3720⟩ (accessed 2019-06-24).

[9] : Data Plane Development Kit, DPDK Project (online), available from ⟨http://dpdk.org/⟩ (accessed 2019-06-24).