Vol.2012-OS-120 No.10
2012/2/29

イーサネットで接続した I/O デバイス間の直接データ移動法

鈴木 順[†] 安田真人[†] 高橋雅彦[†] 飛鷹洋一[††]
樋口淳一[†] 渡邊義和[†] 吉川隆士[†]

ストレージシステムにおけるデータ再配置のためのストレージデバイス間のデータ移動や，Network-Attached Storage に対するクライアントからのデータのバックアップは，I/O デバイス間でデータを移動させる処理の一例である．現在のコンピュータシステムでは，ネットワークインタフェースカード(NIC)や Solid-State Drive (SSD)をはじめとした I/O デバイス間でデータを移動する場合，データを一度ホストサーバのメインメモリに送る必要がある．このため，ホストサーバと I/O デバイスがネットワークで接続されたコンピュータシステムでは，ホストサーバのネットワークリンクが I/O デバイス間のデータ移動のボトルネックとなっていた．本稿では，I/O デバイス間のデータ移動をホストサーバを経由せず直接行うことで，データ移動帯域のボトルネックを解消し，広帯域にデータ移動を行う手法を提案する．2 つの PCI Express (PCIe) SSD を用いて評価を行った結果，データ移動帯域が従来手法より 86%向上し，SSD の最大性能でデータ移動を行うことが出来た．

High-Throughput Direct Data Transfer between I/O Devices

Jun Suzuki[†] Masato Yasuda[†] Masahiko Takahashi[†]
Yoichi Hidaka[††] Junichi Higuchi[†] Yoshikazu Watanabe[†]
and Takashi Yoshikawa[†]

Data transfer between storage devices for data reallocation in a storage system and backing up data from a client to a network-attached storage are examples of data processing that transfers data between I/O devices without modifying data. In current systems, transferred data between I/O devices must be sent once to the main memory of the server hosting the I/O devices. However, in a system which interconnects a host server and I/O devices using a network method, the network bandwidth of the server becomes a bottleneck for data transfer. This paper proposes a method to transfer data directly between I/O devices. Evaluation using two PCI Express (PCIe) solid-state drives (SSDs) showed the proposed method increased the bandwidth of the data transfer by 86% and to the maximum throughput of the SSD.

1. はじめに

サーバコンピュータは，提供するサービスに応じて様々な I/O デバイスを保持する必要がある．これらの I/O デバイスの例として，ネットワークインタフェースカード(NIC)，PCI Express Solid-State Drive (PCIe SSD)，ストレージホストバスアダプタ(HBA)が挙げられる．このため，サーバに必要な I/O デバイスを，サーバの I/O スロット数に制限なく容易な管理で割り当てる技術が必要となる．そのための手法として，複数のサーバと複数の I/O デバイスとをネットワークを用いて接続し，ネットワークの接続を制御することで所望の I/O デバイスを任意のサーバに割り当てる技術が開発されてきた[1][2][3][4][5].

これらの I/O デバイスを用いたデータ処理では, I/O デバイス間でデータをブロックレベルで移動する処理が多く含まれる[6]. 一例として, Network-Attached Storage (NAS) にクライアントからデータをバックアップする場合，データは NAS の NIC で受信され，そのまま NAS のディスクに書き込まれる．また Hadoop では，複数の HDFS ノードにデータのレプリカを作成するため，パイプラインと呼ぶ手法を用いている．この処理では，ある HDFS ノードの NIC で受信されたデータが，再びその NIC から他の HDFS ノードに送信される．

ところが，現在のコンピュータシステムでは，I/O デバイス間でデータを移動する場合，全てのデータをホストサーバのメインメモリに送る必要がある．このため，I/O デバイス間を移動する全てのデータがホストサーバのネットワークリンクを通過することになり，特に I/O デバイス間のデータ移動を並列で行うシステムでは，I/O デバイス間を移動するデータがホストサーバに集中することで，ホストサーバのリンク帯域が I/O デバイス間のデータ移動帯域のボトルネックとなる．さらに，現在のシステムでは I/O デバイス間のデータ移動の制御をユーザプログラムが行うため，サーバのメモリ内でユーザ空間とカーネル空間のメモリコピーが発生し，レイテンシの増大でデータ帯域がさらに低下する．

我々はこれまでに，ユーザ空間とカーネル空間のメモリコピーに関する遅延を低減するため，I/O デバイス間のデータ移動処理をデバイスドライバとして実装し，カーネル空間で処理を行うことでコピーなくデータ移動を行う手法を提案してきた[7]．しかし，I/O デバイス間を移動する全てのデータがホストサーバを経由することによるデータ移動帯域のボトルネックを解決していなかった．

本稿では，PCIe に準拠する I/O デバイス間でホストサーバを経由せず直接データを

[†] NEC システムプラットフォーム研究所
System Platforms Research Laboratories, NEC Corporation
[††] NEC 応用アプライアンス事業部
Applied Appliance Division, NEC Corporation

移動する Direct Connect 技術を提案する．ホストサーバと I/O デバイスは PCIe over Ethernet 技術を用いてイーサネットで接続する．PCIe over Ethernet 機能は PCIe-イーサネットブリッジ LSI に実装した．通常の I/O デバイスは I/O デバイス間で直接データの通信を行うことができないため，提案手法では送信元デバイスとあて先デバイスの Direct Memory Access (DMA)を PCIe-イーサネットブリッジ内のメモリを用いて中継し，I/O デバイス間の直接データ移動を実現する．これにより，ホストサーバを経由せずに I/O デバイス間で直接データを移動する処理が可能となり，広帯域で低レイテンシのデータ移動を行うことが出来る．

提案手法を用いて 2 つの PCIe SSD 間でデータ移動を行うシステムを実装し評価を行った結果，ホストサーバと SSD 間の帯域に制限されない広帯域なデータ移動を実現できた．また，SSD に発行する命令の並列化と多重化を行うことで，SSD の最大性能の帯域でデータ移動が行えることを確認した．

また，現在用いられているシステムに提案手法を容易に導入するため，システムに対する変更を最小限に抑えるように方式設計を行った．データ移動に用いる I/O デバイスは標準の PCIe デバイスであり，デバイスベンダから供給されるデバイスドライバを用いて駆動する．ホストサーバと I/O デバイスの接続に用いるイーサネットも標準のイーサネットである．

以下本稿では，第 2 節で提案手法のアーキテクチャと 2 つの PCIe SSD 間のデータ移動への実装，第 3 節で実装したシステムに対する評価，第 4 節で今後の予定についてそれぞれ述べ，最後に第 5 節でまとめる．

2. Direct Connect のアーキテクチャと実装

2.1 アーキテクチャ

現在のコンピュータシステムでは，2 つの I/O デバイス同士が直接通信を行うことや，デバイス間で直接データを移動することができない．なぜならこれらの I/O デバイスは，ホストサーバのデバイスドライバから制御され，サーバのメインメモリに対してのみ DMA を用いてデータ通信を行うことが出来るからである．そのため，従来のシステムにおいて I/O デバイス間でデータを移動する場合，データをホストサーバのメモリへ一度送る必要があった．本稿で提案する Direct Connect では，この制限を解決するため，PCIe-イーサネットブリッジ内のメモリをホストサーバのメモリ空間にマップし､仲介バッファとして用いることで送信元デバイスとあて先デバイスの DMA を中継し 2 つの I/O デバイス間の直接データ移動を実現する．またこれにより，I/O デバイス間を移動するデータがホストサーバを経由することによる帯域ボトルネックを解消する．

提案する Direct Connect のアーキテクチャを図 1 に示す．以下ではまず，Direct

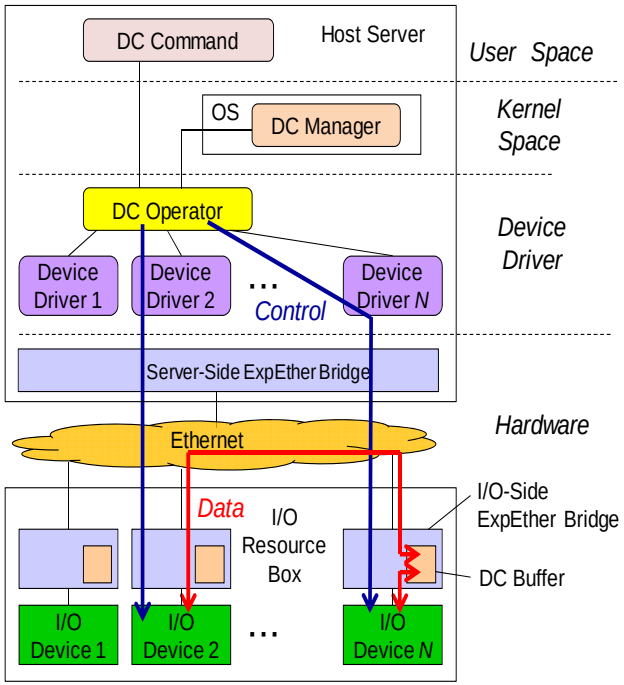

図 1 Direct Connect のアーキテクチャ

Connect のハードウェア構成について述べる．I/O リソースボックスは，I/O デバイスを収容する専用のシャーシである．ボックス内の I/O デバイスはイーサネットを用いてホストサーバに接続される．ExpEther(エクスプレスイーサ)ブリッジは，我々が[1]で提案した PCIe-イーサネットブリッジであり，PCIe のパケットをイーサネットフレームにカプセル化しサーバと I/O デバイスの間を伝送する．ExpEther を用いたシステムでは，サーバの PCIe バスがイーサネットで接続した I/O デバイスまで仮想的に延長される．このため，サーバはイーサネットで接続した I/O デバイスをデバイスドライバの変更なくローカルデバイスと同様に扱うことができる．本稿で提案する手法では，2 つの I/O デバイスの DMA を中継するため，個々の I/O デバイス側の ExpEther ブリッジの内部に Direct Connect (DC)バッファを実装している．DC バッファはホストサーバのメモリ空間にマップされるため，I/O デバイスは DC バッファがマップされたメモリアドレスに DMA を行うことで，DC バッファに対しデータ通信を行うことが出来る．

一方，Direct Connect に関するホストサーバのソフトウェア構成は，DC コマンド，

DC オペレータ, DC マネージャの 3 つのコンポーネントから成る. DC コマンドは I/O デバイス間のデータ移動を要求するユーザプログラムである．DC コマンドは，データを移動する I/O デバイスや，移動するデータのデータ長，I/O デバイスがストレージデバイスの場合は移動するデータの開始アドレス等を指定する．

実際のデータ移動制御は DC オペレータが I/O デバイスのデバイスドライバを用いて行う．2 つの I/O デバイス間でデータ移動を行う場合，DC オペレータはデータの送信元デバイスのデバイスドライバにリード要求を発行し，送信元デバイスのいずれかの DC バッファに対しデータを書き込ませる．続いて，データのあて先デバイスのデバイスドライバにライト要求を発行し，あて先デバイスに DC バッファに書き込まれたデータを読み込ませる．これらの動作により，送信元デバイスとあて先デバイスの DMA が DC バッファで中継され，2 つの I/O デバイス間でホストサーバを経由せずに直接データ移動が行える．ここで I/O デバイス間のデータ移動では，I/O デバイス同士を接続するネットワークの帯域をできるだけ有効に用いて広帯域のデータ移動を行うことが重要である．そこで提案手法では，送信元デバイスとあて先デバイスに対する I/O 命令の発行を並列化し，送信元デバイスからのデータの読み込みとあて先デバイスに対するデータの書き込みを並行に行う．また，1 つの I/O デバイスに対して一度に複数の I/O 命令を発行し，I/O 命令の多重化を行う．この多重化により，ホストサーバと I/O デバイスの通信遅延による I/O 性能へのオーバヘッドを削減し，I/O デバイスの処理帯域を向上させることができる．なお，これらの処理を行う DC オペレータはデバイスドライバとして実装されており, カーネルに対する変更は小さく抑えている．

DC マネージャは，ホストサーバに接続する I/O デバイスのデータベースである．DC マネージャを用いることで，DC コマンドから指定された I/O デバイスの名前に対し，I/O デバイスに対応するカーネル空間内のデバイスデータのアドレスを検索できる．このアドレスは，DC オペレータが I/O デバイス間でデータ移動を行う際に，I/O デバイスのデバイスドライバを呼び出すために必要となる．

2.2 実装

提案手法の評価を行うため, 2 つの PCIe SSD 間で直接データ移動を行うシステムを実装した．ホストサーバと I/O リソースボックスは 2 ホップの 10GbE スイッチを用いて接続した．実装に用いた構成を表 1 に示す．ホストサーバ側の ExpEther ブリッジは図 2 に示すように PCIe I/O カードの実装形態であり，サーバの I/O スロットに挿入して用いる．一方 I/O デバイス側の ExpEther ブリッジは，I/O リソースボックス内の各スロットに個別に実装されている．図 3 に I/O リソースボックスの写真を示す．
DC マネージャはカーネルの I/O スタックに実装し，DC コマンドから sda や sdb などデバイスノードを指定することで, データ移動を行う SSD の領域を指定できるようにした．また，PCIe SSD のデバイスが，その DMA のあて先として DC バッファを指定できるように SSD のデバイスドライバの一部を改造した．ただし，DC バッファに対

表 1　実装に用いた構成

ホストサーバ	OS	CentOS 5.5
	CPU	Intel® Xeon® CPU L5520 4 cores 2 processors
	Memory	8GB
イーサネット		10GbE
SSD		OCZ Z-DRIVE M84 PCE-EXPRESS SSD 256GB

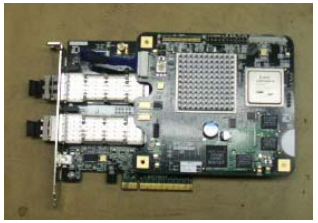

図 2　ホストサーバ側 ExpEther ブリッジ

図 3　I/O リソースボックス

する DMA は Direct Connect 処理のみで使用し，他のアプリケーションが SSD にアクセスする場合，SSD の DMA のあて先は従来と同じサーバのメインメモリとなるようにした．

DC バッファは，ハードウェアの制限から，ExpEther ブリッジとは別の Field Programmable Gate Array (FPGA)カードを用いて実装し，その FPGA カードを I/O リソースボックスに挿入した．FPGA には DC バッファの制御回路を実装し，DC バッファのメモリ領域として FPGA と接続する DDR2 を用いた．制御回路には Altera 社の PCI

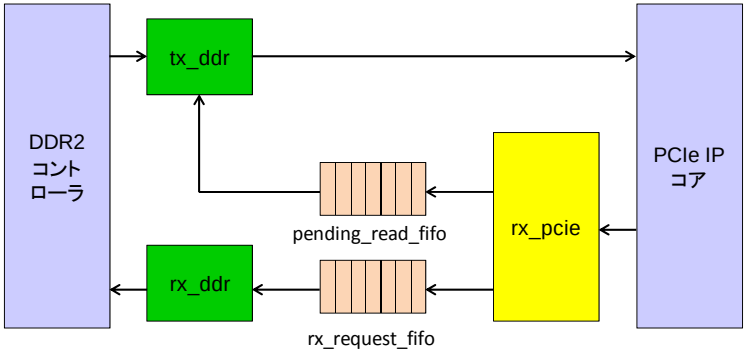

図 4　DC バッファの制御回路のモジュール構成

Express-to-DDR2 SDRAM Reference Design を改造して用いた[8]．実装した制御回路のモジュール構成を図 4 に示す．PCIe IP コアは FPGA カードと I/O リソースボックスを接続する PCIe リンクを終端する．rx_pcie は PCIe IP コアから SSD が発行した DC バッファに対する DMA 要求を受信する．DMA 要求とは具体的には PCIe の Memory Write Request パケットか Memory Read Request パケットである．受信したパケットが Memory Write Request の場合，要求は First-In First-Out (FIFO)キューの rx_request_fifo にキューイングされ，続いて rx_ddr によって順にキューから取りだされ，DDR2 コントローラに対するメモリ書き込み要求に変換される．一方，受信したパケットが Memory Read Request の場合，要求は rx_request_fifo にキューイングされると同時に，別の FIFO である pending_read_fifo にも登録される．これにより， DDR2 から SSD が要求したデータが読みだされた際，tx_ddr はデータを要求した Memory Read Request パケットの情報を pending_read_fifo から取りだし，Completion パケット(応答パケット)のヘッダを作成し PCIe IP コアを通してデータを要求した SSD に返信する．また，PCIe IP コアは，ホストサーバの起動時にホストサーバの物理メモリへのマッピングを要求する．これにより，各 SSD はホストサーバの物理メモリにマップされた DC バッファに対し DMA を行うことができる．

3.　システム評価

システム評価では，2 つの PCIe SSD 間を移動するデータが，サーバに送られることによるボトルネックなく直接広帯域に伝送されることを確認した．

はじめに，PCIe パケットをモニタする測定器である PCIe アナライザを使用し，SSD 間を移動するデータがホストサーバのメインメモリではなく DC バッファを経由して伝送されることを確認した．

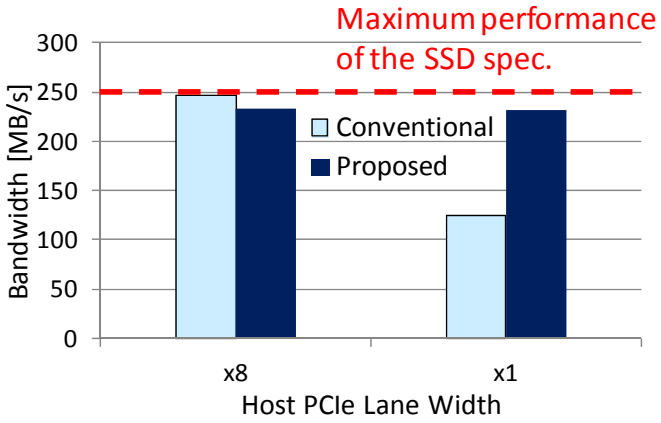

図 5　2 つの PCIe SSD 間のデータ移動帯域

次に，提案手法を用いて 1GB のデータを 2 つの SSD 間で移動する場合のデータ帯域を測定した．図 5 に測定結果を示す．測定は 10 回行いそれらの結果の平均値を取得した．従来手法との比較のため，SSD 間を移動するデータがサーバのメモリへ送られる場合の帯域を従来手法として測定した．実験では提案手法と従来手法について，次の 2 つの環境下でデータ移動帯域を測定した．1 つはホストサーバ側の ExpEther ブリッジの PCIe レーン数を x8 (16Gb/s)に設定した場合である．このときサーバと SSD 間の帯域はデータ移動帯域のボトルネックとならない．もう一方は，サーバ側の ExpEther ブリッジの PCIe レーン数を x1 (2Gb/s)とした場合である．このとき，サーバと SSD 間の帯域がデータ移動帯域のボトルネックとなる．図 5 に示した測定結果を参照すると，ExpEther ブリッジのレーン数が x8 の場合，SSD 間のデータ移動帯域は提案手法と従来手法で同じである．一方 x1 の場合，提案手法のデータ移動帯域は従来手法に対し 86%向上した．ここで実装に用いた SSD の書き込み帯域のスペック値が 250MB/s であることから，従来では帯域ボトルネックが発生する x1 の環境下でも，提案手法では SSD の最大性能でデータ移動が行えることを確認できた．また，PCIe レーン数が x8 の場合の測定結果では，従来手法と提案手法の帯域にわずかの差分がある．この原因として，我々は DC バッファに用いたハードウェアやその設計，DC バッファを別の FPGA カードに実装したことによるイーサネットスイッチのホップ数の増加などを考えている．以上のデータ移動帯域の測定結果より，ホストサーバと I/O デバイスの間の帯域が小さい場合でも，提案手法を用いることで I/O デバイス間では直接広帯域に

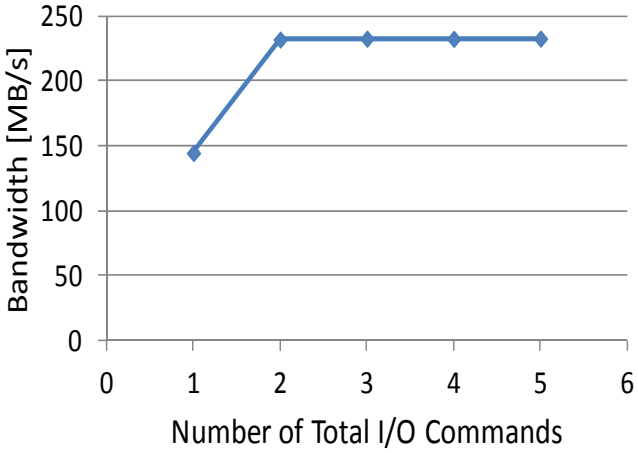

図 6　SSD に発行する合計 I/O 命令数に対するデータ移動帯域

データ移動が行えることを確認できた．

3 つめの評価では，Direct Connect において PCIe SSD に一度に発行する I/O 命令数を変化させてデータ移動帯域の測定を行った．図 6 は送信元 SSD とあて先 SSD に，合わせて一度に発行する I/O 命令数に対するデータ移動帯域である．合計の I/O 命令数を増加させた場合，送信元 SSD からのデータの読み込みとあて先 SSD へのデータの書き込みを並行して行うことができる．また，1 つの SSD に I/O 命令の完了を待たずに発行する I/O 命令数が増加し，I/O 命令を多重化できる．図 6 を参照すると，データ移動帯域は合計 I/O 命令数が 1 から 2 に増加すると約 100MB/s 増大するが，その後は命令数を増加しても帯域が変わらないことがわかる．これは，多重度を 2 に増加することにより，送信元 SSD からのデータの読み込みとあて先 SSD へのデータの書き込みが並列化され，データ移動帯域が増加する一方，命令数が 2 以上の範囲ではあて先 SSD の最大性能でデータ書き込みを行っているため，それ以上 SSD に対する I/O 命令の多重度を増加させても帯域が増加しないためと考えられる．図 6 の実験結果から，提案手法では I/O 命令の多重度を増加させることで，I/O デバイスの最大性能でデータ移動が行えることを確認できた．なお先に図 5 に示した実験結果は多重度を 3 に設定した場合である．今回の評価はシステム内の SSD が送信元 SSD とあて先 SSD の 2 つの場合の結果だが，多数の SSD を用いたシステムでは，SSD 間のデータ移動を複数のペアで並列して行うことができ，従来のシステムに対しより広帯域なデータ移動システムを実現できる．

4.　今後の予定

本稿では提案手法を用いて 2 つの SSD 間で直接広帯域にデータ移動を行うシステムを実現した. 今後, SSD と NIC の間の直接データ移動にも提案手法の適用を検討する．また，2 つの SSD 間で直接データ移動を行う場合，移動するデータに関するファイルシステムのメタデータを制御する技術も重要である．我々は今後，SSD 間で直接データを移動するシステムにおけるメタデータ処理の技術にも取り組む予定である．

5.　まとめ

本稿では 2 つの I/O デバイスの間で直接広帯域にデータ移動を行う手法を提案した．通常の I/O デバイス同士は，直接通信を行うことやデータの交換を行うことができない．このため提案手法では，PCIe-イーサネットブリッジ内に実装したメモリを仲介バッファとして用いることで，データを移動する送信元デバイスとあて先デバイスの DMA を中継し, ホストサーバを経由しない I/O デバイス間の直接データ移動を実現した．

また, 提案手法を用いて 2 つの PCIe SSD 間で直接データ移動を行うシステムを実装し評価を行った．これにより，従来手法ではホストサーバと SSD を接続するネットワーク帯域がボトルネックとなりデータ移動の帯域が低下する環境下でも，提案手法ではボトルネックの影響なく SSD の最大性能でデータ移動が行えることを確認した．

謝辞

本研究に関し多くの有益なコメントをいただいた小比賀亮仁氏，並びに日頃ご指導いただく神谷聡史氏，下司昌幸氏，西原基夫氏をはじめ NEC 関係各位に感謝します．

本研究の一部は，独立行政法人新エネルギー・産業技術総合開発機構(NEDO)の委託事業による成果である．

参考文献

1)　J. Suzuki, Y. Hidaka, J. Higuchi, T. Yoshikawa, and A. Iwata, "ExpressEther - Ethernet-Based Virtualization Technology for Reconfigurable Hardware Platform," 14th IEEE Symposium on High-Performance Interconnects (HOTI'06), pp. 45-51, 2006.

2)　J. Suzuki, Y. Hidaka, J. Higuchi, T. Baba, N. Kami, and T. Yoshikawa, “Multi-Root Share of Single-Root I/O Virtualization (SR-IOV) Compliant PCI,” 18th Annual Symposium on High-Performance Interconnects (HOTI 18), 2010.

3)　HP Virtual Connect Technology: http://h20195.www2.hp.com/v2/GetPDF.aspx/4AA0-5821ENW.pdf.

Vol.2012-OS-120 No.10
2012/2/29

4) P. Kirkpatrick, A. Alsaadi, P. Mididuddi, P. Chauhan, A. Daghi, D. Kim, S. Kim, K. R. Kishore, P. Kulkarni, M. Lyons, K. Malwankar, H. Ravi, S. Saikumar, M. Subramanian, M. Thangaraj, A. Vasudev, V. Venkataraghavan, C. Yang, and W. Yung, “Flash Memory Performance on a Highly Scalable IOV System,” 3rd Workshop on I/O Virtualization (WIOV’11), 2011.
5) V. Krishnan, T. Comins, R. Stalzer, and D. Wong, “A Case Study in I/O Disaggregation using PCI Express Advanced Switching Interconnect,” 14th IEEE Symposium on High-Performance Interconnects (HOTI'06), 2006.
6) B. Rhoden, K. Klues, D. Zhu, and E. Brewer, “Improving Per-Node Efficiency in the Datacenter with New OS Abstractions,” Symp. on Cloud Computing (SOCC’11), 2011.
7) 鈴木順，高橋雅彦，飛鷹洋一，馬場輝幸，菅原智義，吉川隆士，“Solid State Drive 間の高速データ移動法，” 2011 年 4 月 OS 研究会(第 117 回システムソフトウェアとオペレーティング・システム研究会)，2011.
8) “PCI Express-to-DDR2 SDRAM Reference Design,” Altera Application Note 431, 2006, ver. 1.0. http://www.altera.com/literature/an/an431.pdf