Vol.2010-OS-113 No.7
2010/1/28

# イーサネット接続 SSD による コンピュータのメモリ拡張

鈴木　順[†]　馬場輝幸[†]　飛鷹洋一[††]　樋口淳一[†]
加美伸治[†]　内田智士[†]　高橋雅彦[†]
菅原智義[†]　吉川隆士[†]

クラウドコンピューティングでは、サービスの実施状況にあわせた性能スケールアップが求められる。しかし、計算システムの性能を大きく左右するメモリは各コンピュータに個別に搭載されている以上の拡張は困難である。本稿では、PCI Express (PC Ie)にフラッシュメモリコントローラを直結した PCIe 型 Solid S tate Disk (SSD)を我々の開発した PCIe over Ethernet 技術、ExpEther でネットワーク接続し、これを仮想メモリに割当てる事でシステムメモリの拡張を実現した。SSD へはイーサネット越しに Direct Memory Access (DMA)による高速低遅延データ接続が行え、これをスワップデバイスとして用いることで、ソフトウェアとハードウェアとも市販のままで簡便にメモリ拡張が可能となる。これを用いて実機により、データベースのベンチマークで 2GB のローカルメモリに対して 10GB の DB サイズを取っても十分なシステム動作が得られる事を確認した。

# Computer Memory Expansion with SSD Attached via Ethernet

Jun Suzuki[†]　Teruyuki Baba[†]　Yoichi Hidaka[††]
Junichi Higuchi[†]　Nobuharu Kami[†]
Satoshi Uchida[†]　Masahiko Takahashi[†]
Tomoyoshi Sugawara[†]　and　Takashi Yoshikawa[†]

For cloud computing, co mputer in frastructures n eed to pr ovide co mputing resources adaptively, in accord w ith the resource ut ilization. Resources in clude, ho wever, lo cal memories that cannot exceed the amount loaded to each computer. We present a method for ad aptively a ttaching a PCI-Express-based sol id state d isk (SSD) to a computer and expand its local memory using virtual memory system. We use PCI Express (PCIe) over Ethernet technology “ExpEther”, to interconnect a co mputer and an SSD via a standard Ethernet. The d ata transfer bet ween the local memory of th e co mputer and the SSD is performed wi th di rect memory ac cess (DM A). Assigning a n SSD to a c omputer as a swap device allows the local memory of a computer to be expanded without any change to cu rrent softwar e or h ardware. With ou r proposed method, we are able to ach ieve a database benchmark output maintaining its performance even when 10-GB database size is loaded to a 2-GB local memory system.

## 1. はじめに

クラウドコンピューティングでは、コンピューティング資源をサービスの負荷に応じて柔軟に割り当てる必要がある。しかし、コンピュータのメモリは各コンピュータに個別に搭載され、コンピュータ間での共有ができない。このためコンピュータへのメモリ資源の割り当ては、そのピーク時の使用率を考慮して行っており、コンピュータの大半の運用時間において稼働しないメモリが生じている。また、コンピュータのメモリは各コンピュータに個別に割り当てられるため、アプリケーションはコンピュータに搭載されているメモリを超えて消費することができない。このため、コンピュータに実装されるメモリスロットがシステムに搭載可能なメモリの制限となっている。この制限は、今日、CPU ソケットあたりの CPU コア数の増加がメモリ密度の増加を上回っていることと、仮想化技術の出現により 1 コアあたりのメモリ容量の要求が増加したことで、益々顕在化している問題となっている[1]。

コンピュータのメモリに関するこれらの課題は、コンピュータの外部から適応的にメモリ資源を割り当てることが解決手段のひとつであると考えられる。これにより、コンピュータに個別に搭載されているメモリを超えた拡張が可能となる。また、アプリケーションの負荷に応じてメモリ資源を割り当てることで、メモリの稼働率を高めることもできる。

これまでもコンピュータの外部からメモリ資源を割り当てる技術がいくつか提案されてきた。各コンピュータのローカルメモリを共有する方式では、リモートページング[2, 3, 4, 5, 6] や、複数のコンピュータ全体でグローバルにメモリ管理を行う方法[7, 8, 9] が提案されている。また、特別なメモリモジュールを作成し、第 2 層目のシステムメモリとしてコンピュータの外部から割り当てる方式も提案された[1, 10, 11 , 12]。始めの 2 つの方式では、ソフトウェアで行うメモリ管理と TCP/IP によるメモリ格納データの通信が、メモリシステムのボトルネックとなっている。また、メモリモジュールを作成する方式では、コンピュータのローカルメモリと外部から割り当てられたメモリが階層構造を形成するため、ソフトウェアとハードウェアの両方で階層構造への対応が必要となる欠点がある。

一方、近年フラッシュメモリがコンピュータのローカルメモリとハードディスクド

[†] NEC システムプラットフォーム研究所
System Platforms Research Laboratories, NEC Corporation
[††] NEC IP ネットワーク事業部
IP Network Division, NEC Corporation

ライブ(HDD)を補完する構成要素として注目を集めている[13, 14, 15]。我々は本稿で、フラッシュメモリを用いてコンピュータのメモリを高速かつ既存のプラットフォームへの変更なく拡張する方式を提案する。フラッシュメモリは、DRAM を用いるコンピュータのローカルメモリより消費電力が低く、速度は DRAM に劣るものの HDD より速い。このフラッシュメモリを、コンピュータに組み込む良い方法の 1 つがストレージのインターフェースを介さずに PCI Expres s へフラッシュメモリコントローラを直結する PCI Express (PCIe)型 Solid State Disk (SSD)である。この種のデバイスでは、デバイスドライバのスタック内でブロックデバイスの挙動を疑似することで、SSD がソフトウェアからブロックデバイスとして認識される[16]。

提案方式では、PCIe 型 SSD を用いてコンピュータのメモリを標準イーサネットで拡張する。SSD の接続には、鈴木らが提案した、PCIe スイッチをイーサネット上で分散仮想的に構成する形の PCIe over Ethernt 技術である ExpEther (Ex press Ether)を用いる[17]。ローカルメモリと SSD は、Direct Memory Access (DMA)を用いて高速にデータ通信を行う。接続した SSD をスワップデバイスとして用いることで、ソフトウェアとハードウェアの変更なくメモリ拡張が可能となる。

本稿では、第 2 節でメモリ拡張方式の要件を議論する。次に、第 3 節でイーサネット接続 SSD によるメモリ拡張方式を提案し、第 4 節で提案手法を用いて行った実験の結果を述べる。最後に第 5 節でまとめる。

## 2. メモリ拡張方式の要件

本稿で提案する、ネットワークを用いたメモリ拡張方式に対するシステム要件を以下に整理する。

- **性能:** 高いメモリシステム性能を実現するため、コンピュータのメモリと SSD は、広帯域、低遅延の通信を実現できる手段で接続しなければならない。
- **動的で柔軟な資源割り当て:** アプリケーションの負荷に応じて適応的にメモリ資源を割り当てるため、SSD を所望のコンピュータに、そのサービスを停止させることなく割り当てられなければならない。
- **メモリ性能の相違を意識したデータ配置:** SSD はコンピュータのローカルメモリと比較して性能が 1 オーダー以上異なる。このため、システムソフトウェアはこの性能差を意識したデータ配置を行い、アプリケーションに対するシステム性能を最適化しなければならない。
- **汎用コンポーネントの利用:** メモリ拡張方式は、一般に用いられているシステムと親和性を持って動作しなければ導入への障壁が高く、産業的な意義が低い。このマイグレーションの容易化のため、既存システムへの変更を最小限に抑えるとともに、他のシステムで用いている汎用コンポーネントを変更なく利用できなければならない。

## 3. イーサネット接続 SSD によるメモリ拡張

我々は、以下に示す 3 つの技術を取り入れたメモリ拡張方式を提案し 2 節で述べたシステム要件を満たす(図 1)。

- **ネットワーク上の DMA:** コンピュータのローカルメモリと SSD のネットワーク越しのデータ通信を、DMA を用いて行う事を実現し、これにより、広帯域・低遅延な接続を実現する。
- **ホットプラグ・リムーブ:** ネットワーク越しに SSD のホットプラグおよびリムーブを実現し、コンピュータのサービスを停止させることなく SSD 資源を所望のコンピュータに適応的に割り当てる。
- **ページスワップ:** メモリ拡張システムにおけるデータ配置の最適化を今日のシステムソフトウェアへの影響なく実現するため、コンピュータに接続した SSD を OS の制御する仮想メモリ（スワップデバイス）として用いる。

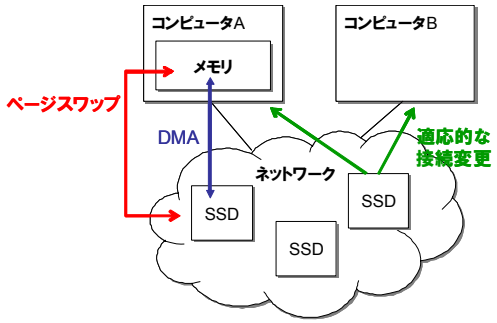



図 1 メモリ拡張のための 3 つの技術

以下、本節ではこれら 3 つの技術の詳細を述べる。

### 3.1 ネットワーク上の DMA

提案手法では、コンピュータのローカルメモリと SSD の間で高速なデータ通信を実現するため、イーサネット上で PCIe に準拠する DMA を行う。DMA により、ソフトウェアによるプロトコル処理や通信データのメモリコピーを伴うことなくデータ通信を行うことができる。これにより、ストレージ資源をネットワークで共有する従来のプロトコルと比較して高速なデータ通信が実現できる。

提案手法では、イーサネット上で PCIe に準拠した DMA を行うため、以下に述べる ExpEther の 2 つの機能を用いる(図 2)。(1)ExpEther は、コンピュータの PCIe ツリーを

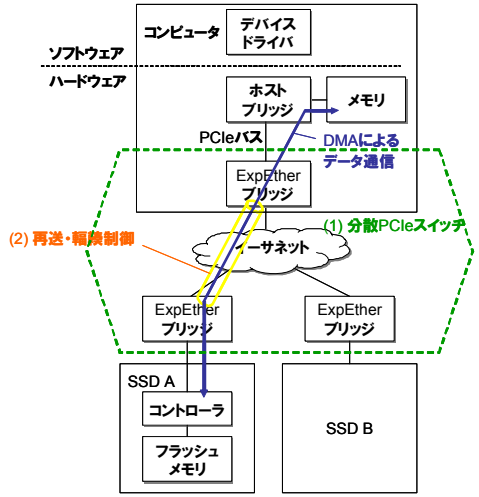



図 2　ExpEther の分散 PCIe スイッチと再送・輻輳制御

イーサネットに展開し、PCIe スイッチとして動作することができる。(2)ExpEther は、イーサネットフレームにカプセル化された PCIe パケットをパケットロスなく低遅延で伝送することができる[18 ]。以下、これら 2 つの機能について述べる。

### 3.1.1 PCIe ツリーのイーサネットへの展開

一般に、PCIe スイッチとはコンピュータの CPU/メモリへ複数の I/O デバイスを接続するために、PCIe バスをツリー状に展開するデバイスである。そのため PCIe スイッチは、内部に 1 つの上流 PCI-PCI ブリッジと複数の下流 PCI-PCI ブリッジを持つ構成で実装されている。

ExpEther はこの上流、下流の PCI ブリッジを分離して、CPU/メモリ側と I/O 側に配置し､それぞれをイーサネットで接続することで分散仮想的に PCI Express スイッチと等価な機能を実現する。提案手法では I/O エンドポイントは SSD である。コンピュータの OS からは、コンピュータ側の ExpEther ブリッジは PCIe スイッチの上流 PCI-PCI ブリッジに、SSD 側の ExpEther ブリッジは下流 PCI-PCI ブリッジに見える。通常の OS は、PCIe スイッチのデバイスドライバを備えているため、ExpEther は新たなデバイスドライバを追加することなくコンピュータの PCIe ツリーをイーサネットに拡張することができる。

PCIe ツリーを SSD に延長することで、SSD はコンピュータのアドレス空間にマップされ、コンピュータの I/O スロットに挿入された通常の I/O デバイスと同様に PCIe に準拠する DMA を行うことが可能となる。SSD の DMA は、コンピュータ上の SSD のデバイスドライバが、SSD 上のコントローラを呼び出すことで行われる。

### 3.1.2 PCIe パケットの高信頼低遅延伝送

ExpEther では、ExpEther ブリッジを用いてエンドーエンド間で PCIe パケットをカプセル化したイーサネットフレームの再送・輻輳制御を行う。これにより、汎用イーサネットスイッチを用いながら、PCIe パケットをパケットロスなく低遅延で伝送することができる。

PCIe は通常、信頼性の高い I/O バス機能を提供する。このため、コンピュータシステムは PCIe バスでの I/O パケットの伝送中に、パケットロスが発生しないことを想定している。一方、標準のイーサネットはイーサネットフレームのあて先への到達を保障しない。このため、ExpEther レイヤでロスのないパケット伝送機能を提供することが必要となる。

イーサネットでのパケットロスの主要因は輻輳によるスイッチでのバッファあふれ、パケット廃棄で生じる。ExpEther の輻輳制御機能は以下の通りである。ExpEther は、遅延に基づいた送信レート制御を行う。ExpEther ブリッジは、ブリッジ間の Round Trip Time (RTT)をモニタし、RTT が最小となるように送信レートを調整する。これにより、中継イーサネットスイッチのパケットバッファにおける転送待ち時間を最小化し、ExpEther ブリッジ間でイーサネットフレームを転送待ちなく最小遅延で伝送することができる。一般に PCIe では、DMA 性能が PCIe バスの遅延に大きく依存する。このため、ExpEther の輻輳制御機能は輻輳により生じる伝送遅延を抑制することで、PCIe の性能を最大化する。ここで ExpEther により得られる伝送遅延は、ロスに基づく送信レート制御を行う TCP の伝送遅延と比較して小さい[19]。これは、TCP ではネットワークに輻輳が起こり、パケットロスが発生するまで送信レートを増加させるためである。この結果、TCP ではネットワークスイッチのパケットバッファにおいてパケット転送待ち時間が発生する。

それでも、輻輳や、その他何らかの原因でパケットロスした場合にも、ExpEther の再送機能では､伝送中に欠落したイーサネットフレームを再送することで PCIe パケットがカプセル化されたイーサネットフレームの到達を保障する。

## 3.2 ホットプラグ・リムーブ

SSD 資源をコンピュータのサービスを停止させることなく適応性、拡張性を持って所望のコンピュータに接続するため、ネットワーク越しの I/O デバイスのホットプラグおよびリムーブを実現した。

PCIe プロトコルはコンピュータ内の I/O バスの規格を標準化したものであり、提案手法が対象とする、複数のコンピュータを含むシステム全体の I/O デバイス資源を管理する機能がない。そこで ExpEther では、PCIe に準拠したホットプラグおよびリムーブ機能を、イーサネットを経由して複数のコンピュータに I/O デバイスを割り当てられるよう拡張した。この I/O デバイスの資源管理はマネジメント機構を用いてリモ

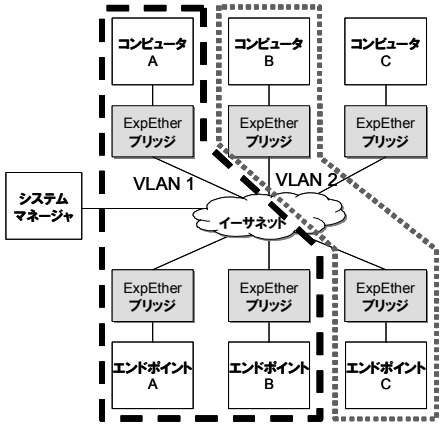



図 3　グループ ID(VLAN)による I/O デバイスのグルーピング

―トから行うことも可能である。以下に詳細を説明する。

まずコンピュータと I/O デバイスの接続であるが、I/O デバイスが接続するコンピュータは、I/O デバイス側の ExpEther ブリッジのグループ ID 番号で決定する。図 3 に示すように、同一のコンピュータに接続する I/O デバイスの ExpEther ブリッジは、接続するコンピュータのグループ ID 番号でグルーピングされる。I/O デバイスの接続を変更する場合、システムマネージャーが I/O デバイス側の ExpEther ブリッジが内部レジスタに保持しているグループ ID 番号を変更する。

また、I/O デバイスのコンピュータへの割り当て処理をトリガするため、ExpEther リッジは同一のグループ ID のグループ内に定期的にインフォメーションフレームをブロードキャストする。このフレームは、コンピュータと I/O デバイスとの間の ExpEther パスのキープアライブ機能に用いられる。コンピュータ側の ExpEther ブリッジが、接続していない I/O デバイス側のブリッジからインフォメーションフレームを受信した場合、コンピュータ側のブリッジはコンピュータに割り込みを上げ、PCIe の標準に準拠したホットプラグ処理を開始する。一方、コンピュータ側の ExpEther ブリッジが接続する I/O デバイス側の ExpEther ブリッジから一定時間インフォメーションフレームを受信しない場合、コンピュータ側のブリッジはコンピュータに割り込みを上げ、ホットリムーブ処理を行う。

次にシステム監視についてであるが、前述したインフォメーションフレームは、システム監視の用途でも用いられる。システムマネージャーは、インフォメーションフレームが保持する情報を用いてシステム内のコンピュータと I/O デバイスのリストを作成する。I/O デバイス側のブリッジが発行するインフォメーションフレームは、接続しているコンピュータ、I/O デバイスのレジスタから取得したデバイス ID およびベンダ ID などの情報を保持している。一方、コンピュータ側のブリッジが発行するフレームは、接続している I/O デバイスの一覧などを保持している。

### 3.3 ページスワップ

現在 SSD は高速、低遅延化が進んでいるが、コンピュータのローカルメモリと比較すると、1 オーダー以上性能が低い。このような環境化でただ単に SSD をメモリマップしてしまうとシステム全体の性能劣化が起きるという課題がある。このため。システムソフトウェアはアクセス性能の違いを考慮してローカルメモリと SSD の間のデータ配置を行い、システム全体の性能を高める必要がある。

そこで提案手法では、OS が既に実装しているスワップ機能を用いてイーサネットで接続した SSD をスワップデバイスとして用いる。これにより、SSD が保持するページがアクセスされると、該当ページがコンピュータのローカルメモリにスワップインされ、最も前にアクセスされたページがスワップアウトされる。これらのスワップ処理に伴うローカルメモリと SSD の間のデータ通信は、前述のとおりイーサネット上の DMA を用いて行われるため、ローカルに置いた SSD デバイスへのアクセスと同等の性能を発揮する。

## 4. 実験結果と考察

我々は、提案手法について、ExpEther を FPGA を用いて実装するとともに、市販 PCIe 型高速 SSD を用いて実機検証すべく以下に述べる 3 つの実験を行った。1 つめの実験では、提案手法を用いてイーサネットで接続した SSD と、従来のストレージ資源をネットワークで共有するプロトコルで接続した SSD のブロック I/O 性能を比較し、提案手法が従来手法と比べて高速なスワップデバイスを提供することを示した。

2 つめの実験では、提案手法を用いることでアプリケーションがローカルメモリの容量を超えてメモリを消費しても、アプリケーションの性能を維持できることを示した。また、提案手法によりコンピュータに個別に搭載するローカルメモリの容量を削減できることを示した。実験にはリレーショナルデータベース(RDB)のベンチマークツールを使用し、RDB ファイルをコンピュータのローカルメモリに配置して行った。

3 つめの実験では、提案手法を用いることでコンピュータに対し SSD 資源を柔軟性と拡張性を持って接続できることを示した。

初めの 2 つの実験は、次の 4 つの構成を用いて行った(図 4)。(a)スワップデバイスとして用いる SSD をコンピュータの I/O スロットに挿入した。(b)コンピュータと SSD を ExpEther (EE)ブリッジを用いて接続した。(c)iSCSI を用いてコンピュータと SSD を接続した。このとき、ターゲットの I/O スロットに SSD を挿入し、イニシエータとターゲットの接続には TCP オフロードエンジン(TOE)付き NIC を用いた。(d)通常の NIC

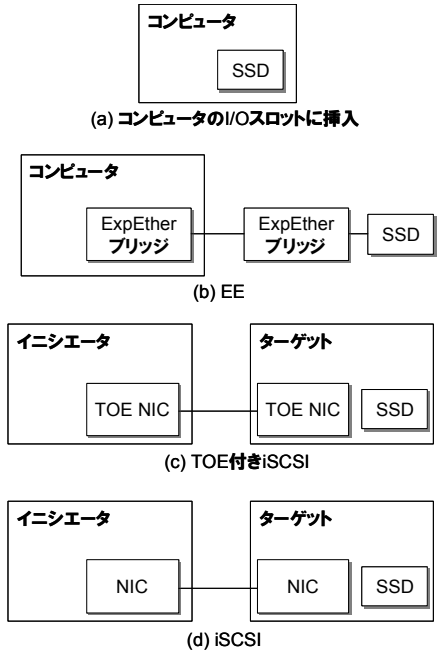



図 4　実験構成

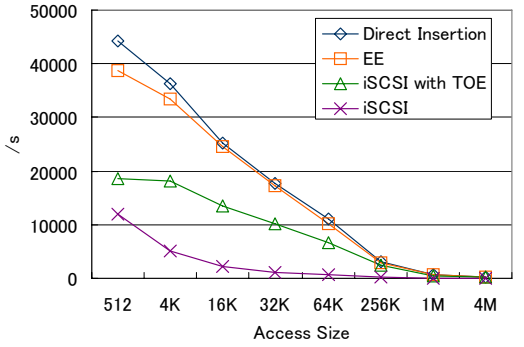
表 1　実験で用いたコンピュータの特製

| 実験名 | ブロック I/O 性能 | データベース |
|---|---|---|
| CPU | Intel Core 2 | Intel Core 2 Quad |
| メモリ | 3 GB | 2, 4, 8 GB |
| OS | Cent OS 5.3 | |
| イーサネット | 10GbE | |
| SSD | 160GB Fusion IO [16]. Write 性能向上モード(容量 50%). | |

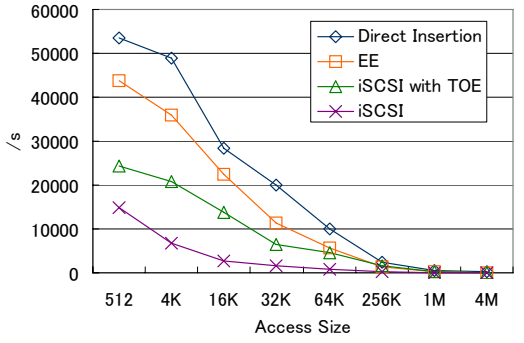
を用いて iSCSI でイニシエータとターゲットを接続した。構成(a)は、コンピュータの I/O スロットに挿入した SSD による DMA 性能を、各実験のベースラインとして用いるために測定した。構成(c)および(d)は、ストレージ資源をネットワークで共有する従来のプロトコルとして比較するために測定した。
初めの 2 つの実験で用いたコンピュータの特性を表 1 に示す。ここでコンピュータ側と SSD 側の ExpEther ブリッジのパケット転送遅延の合計は 1.45 μs だった。また、実験で用いた 10GbE スイッチのレイテンシは 0.92 μs だった。今回、ExpEther ブリッジのプロトタイプは XILINX 社の Vertex V シリーズの FPGA(Field Programmable Gate Array)を用いて実装した。

### 4.1 ブロック I/O 性能

SSD のブロック I/O 性能の測定は、iometer を用いて行った[20]。図 4 に示した 4 つの構成に対して測定した IOPS (I/O per second)を図 5 に示す。グラフの各測定点の測定時間は 10 分である。ここで、スワップ処理で使用される 4KB アクセスの性能を表 2




(a) Random Read IOPS




(b) Random Write IOPS

図 5　ブロック I/O 性能 (a) ランダム Read IOPS (b)ランダム Write IOPS

にまとめる。表 2 には、構成(b)-(d)において、コンピュータと SSD の間に標準イーサネットスイッチを挿入した場合の性能も示す。このとき、イーサネットスイッチの遅延がコンピュータと SSD の間のバスに加算される。各性能は、SSD を I/O スロットに挿入した場合の性能に対し規格化した。実験の結果、提案手法により得られた Read 性能は、I/O スロット挿入時と大きな相違がなく(10%以下)、TOE 付き iSCSI の 2 倍であった。一方 Write 性能は、I/O スロットに挿入した場合と比較して約 30%劣化したが、このときも TOE 付き iSCSI の 2 倍であった。

ここで、Write 性能において測定されたオーバーヘッドを解析するため、コンピュータと SSD の間で PCIe トラフィックをモニタした(図 6 参照)。その結果、Write アクセスでは SSD の DMA コントローラが 4 つまでメモリリクエストを発行し、次のリクエスト発行は前回のリクエストに対する応答を受信するまで待つことがわかった。ここで、これら 4 つのリクエストは並行して発行されるが、コンピュータと SSD の間の遅延が増大すると、リクエスト間で待ち時間が発生する。この問題は、DMA コントローラが一度に発行するリクエスト数を増加し待ち時間を減らすことで解決できると考えられる。

本節で述べたブロック I/O 性能の結果は、提案手法がコンピュータのローカルメモリと SSD の間のデータ通信を DMA を用いて行うことで、高速なスワップデバイスを提供できることを示している。またその性能は、ストレージ資源をネットワークで共有する従来のプロトコルの性能より優れている。これは、従来のプロトコルがコンピュータのメモリと SSD の間でデータ通信を行う際、プロトコル処理とメモリコピーをソフトウェアで行っているのに対して、本方式はアーキテクチャ的な工夫によりすべてハードウェアでの実装を実現できているためである。

表 2　4KB アクセス時の IOPS. 性能は I/O スロット挿入時に対し規格化

| Direct | Insertion | EE | iSCSI w/ TOE | iSCSI |
|---|---|---|---|---|
| Ran. Read | 100 | 92 | 50 | 14 |
| Ran. Write | 100 | 74 | 42 | 14 |
| Ran. Read w/ Switch | 100 | 91 | 46 | 14 |
| Ran. Write w/ Switch | 100 | 68 | 39 | 14 |

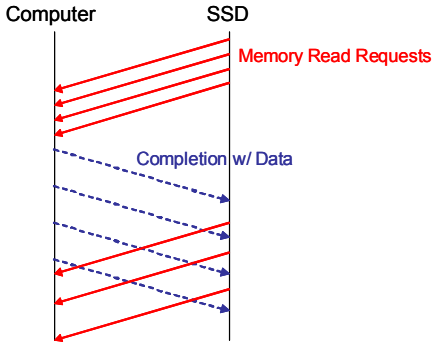



図 6　コンピュータと SSD 間の PCIe トラフィック

### 4.2　メモリ拡張

本節では、提案手法によりアプリケーションがコンピュータに搭載されている以上のメモリを消費した場合でも、その性能を維持できることを示す。また、コンピュータのローカルメモリに要求される容量の一部を SSD に移すことで、個別のコンピュータに搭載するローカルメモリを削減できることを示す。実験は、RDB のベンチマークツールを用いて行った。また、RDB ファイルはコンピュータのメモリ上に配置した。

RDB のプラットフォームには、postgresql を用いた。また、ベンチマークツールは TPC-B に類似する pgbench を用いた[21]。ベンチマークと用いたパラメータを表 3 にまとめる。

実験では、RDB ファイルをメモリ上に仮想的に作成したディスクであるラムディスクに配置し、ファイルサイズを増加させた。このとき、ファイルサイズがコンピュータのローカルメモリのサイズを超えると、その一部がスワップデバイスにスワップアウトされる。実験は、図 4 に示した 4 つの構成に対して行った。ここで(c)の構成(TOE 付き iSCSI)では、ソフトウェアのバグのためスワップ機能が動作し始めるとシステムがストールし、測定を行うことができなかった。また、図 4 の 4 つの構成とは別に、性能を評価するためのベースラインとして、コンピュータが内蔵する HDD をスワップデバイスとして用いた場合の性能も測定した。

初めに、提案手法によるコンピュータのメモリ拡張と削減の効果を測定した。図 7 は、データベース(DB)ファイルの容量を変化させたときの提案手法とベースラインの性能である。ここで提案手法に対する測定では、コンピュータのローカルメモリのサイズを 4GB と 2GB に設定し、イーサネットで接続した SSD をスワップデバイスとして使用した。一方、ベースラインに対する測定では、メモリサイズを 8, 4, 2GB とし、コンピュータが内蔵する HDD をスワップデバイスとして用いた。

表 3　RDB ベンチマークツールと用いたパラメータ

| RDB post | gresql 8.1 |
|---|---|
| Benchmark pgbench | (TPC-B-like) |
| #Clients 100 | |
| Transaction per Clients | 1000 |

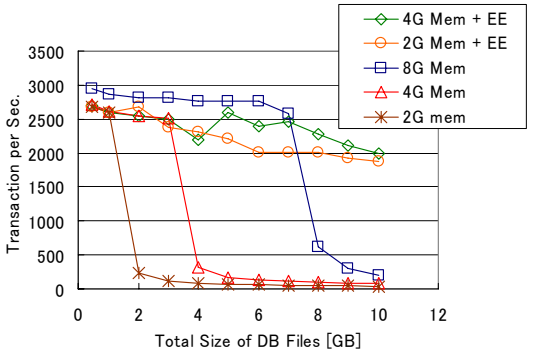



図 7　pgbench 性能. DB はデータベース.

実験の結果、ベースラインでは HDD のアクセス速度が遅いため、HDD へのスワップアウトが始まると性能が急速に低下した。一方提案手法では、2GB と 4GB の両方で SSD へのスワップアウトが開始されても性能低下が小さかった。10GB の DB ファイルの結果に着目すると、ベースラインでは全ての場合において性能が低下している半面、提案手法では、アプリケーションがローカルメモリ容量を超えるメモリを消費した場合でも性能が維持できていることがわかる。また、6GB の DB ファイルの結果に着目すると、提案手法は、全てのデータがローカルメモリ上にある 8GB メモリのベースラインと比較して、性能劣化が 4GB のローカルメモリで 13%、2GB のローカルメモリで 27%に抑えられた。これらの結果は、提案手法を用いることでローカルメモリに要求される容量の 1/3 を僅か 13%のオーバーヘッドで、2/3 を僅か 27%のオーバーヘッドで削減できることを示している。

また、提案手法によるメモリ拡張の効果はコンピュータの CPU 使用率からも確認できた。図 8 は、図 7 の各測定におけるコンピュータの CPU 使用率である。ここで 10GB の DB ファイルの結果に着目すると、8GB のメモリを保持するベースラインは大半の CPU 時間をスワップ処理による I/O 待ちに消費している。一方、提案手法はアプリケーションの処理が維持できていることがわかる。また、6GB の DB ファイルの結果は、提案手法によるコンピュータのローカルメモリ削減の負荷は、1/3 のローカルメモリで 31%、2/3 のローカルメモリで 30%であることを示している。

次に、提案手法とネットワークでストレージ資源を共有する従来のプロトコルとの性能を比較した。図 9 は、提案手法と、従来プロトコルとして iSCSI を用いた場合の結果である。10GB の DB ファイルでは、提案手法が iSCSI より 4GB のローカルメモリで 110%、2GB で 139%性能が優れていた。これらの性能差は、データ通信の方式の違いによるものであると考えられる。すなわち、提案手法は、イーサネット上の DMA で高速にデータ通信を行う。一方 iSCSI では、ソフトウェアによるプロトコル処理やメモリコピーを伴う。この差が性能に効いたと思われる。また、本実験では前述した

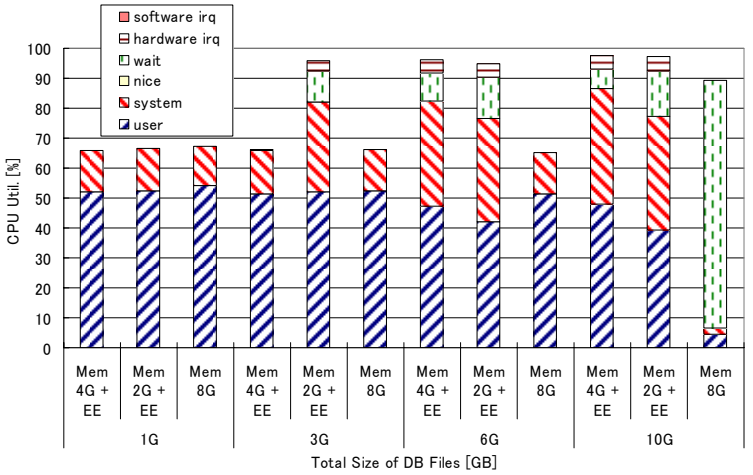



図 8　pgbench 実行時の CPU 使用率

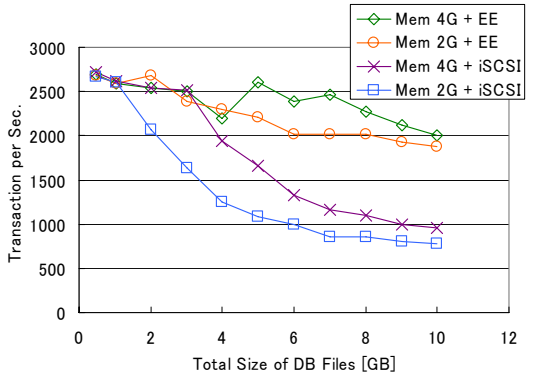



図 9　提案手法と従来手法の pgbench 性能の比較

ようにTOE付きiSCSIの結果を得ることができなかったが、図5に示したブロックI/O性能から、提案手法はTOE付きiSCSIに対しても性能が優れていることが予想される。

### 4.3 SSD接続の拡張性と柔軟性

4.1節と4.2節で報告した実験は、1つのコンピュータと1つのSSDを接続して行った。しかし、提案手法は標準のイーサネットを用いて、コンピュータにSSDを拡張性と柔軟性を持って接続することが可能である。図10は、これをコンピュータのPCIeツリーを示すlinuxのlspciコマンドの出力で確かめたものである。このコンピュータには、提案手法を用いて3つのSSDが接続した状態である事か表示されている。SSDのコンピュータへの接続は、イーサネット上のシステムマネージャーを用いて前述のグループIDにより適応的に行うことができる。ここで"Unknown d evice."として出力されたのは、デバイス情報がOSに登録されていないためで異常を示すものではない。

```
[root@****** ~]# lspci –vt
-[0000:00]-+-00.0  Intel Corporation 82X38/X48 Express DRAM Controller
           +-01.0-[0000:01-05]--00.0-[0000:02-05]--+-00.0-[0000:03]---00.0  Unknown device 1aed:1005
           |                                       +-01.0-[0000:04]---00.0  Unknown device 1aed:1005
           |                                       \-02.0-[0000:05]---00.0  Unknown device 1aed:1003
           +-06.0-[0000:06]—
           +-19.0  Intel Corporation 82566DM-2 Gigabit Network Connection
           +-1a.0  Intel Corporation 82801I (ICH9 Family) USB UHCI Controller #4
           +-1a.1  Intel Corporation 82801I (ICH9 Family) USB UHCI Controller #5
           +-1a.2  Intel Corporation 82801I (ICH9 Family) USB UHCI Controller #6
```

3つのSSD

図10　3つのSSDを接続したコンピュータのPCIeツリー

## 5. まとめ

本稿では、イーサネットを用いてSSDをコンピュータに接続することにより、コンピュータのメモリを拡張する方式を提案した。提案手法は次の3つの特徴を持つ。(a)ローカルメモリとSSDの間でDMAを用いて高速にデータ通信を行う。(b)ホットプラグおよびリムーブ機能を用いてSSD資源を拡張性と柔軟性を持ってコンピュータに接続する。(c)OSに標準で実装されているページスワップ機能を用いてイーサネットで接続されたSSDを使用し、ローカルメモリとSSDのデータ配置を最適化する。これらの手法により、提案方式を用いることでアプリケーション性能を維持し、汎用コンポーネントを用いてコンピュータのメモリ拡張を行うことができる。

実験で得られたブロックI/O性能の結果は、提案手法がネットワークでストレージ資源を共有する従来のプロトコルより高速なスワップデバイスを提供することを示した。また、データベースを用いたシステム性能評価の結果は、提案手法のイーサネット接続SSDによるメモリ拡張で、アプリケーションがコンピュータのローカルメモリを超えるメモリ容量を消費した場合でも、アプリケーション性能が維持できることを示した。さらに、提案手法はコンピュータに搭載するローカルメモリの1/3を削減しても87%のアプリケーション性能を、2/3を削減しても73%の性能を維持できることを示した。これらの実験結果から、従来は困難であったメモリ拡張が簡便に実現可能である事を実機にて実証した。

**謝辞**　本研究に関し、多くの有益なコメントをいただいた長谷部賀洋氏、島本裕志氏、江頭一廣氏、並びに日頃ご指導頂く矢野由美子氏、加納敏明氏をはじめNEC関係各位に感謝します。また、実験データ取得に協力して頂いた藤田マルセロ幸太氏に感謝します。

本研究の一部は、総務省の委託研究「次世代バックボーンに関する研究開発」プロジェクトの成果である。

## 参考文献

1) K. Lim et al., ISCA-36, pages 267-278, 2009.
2) D. Comer and J. Griffioen, "A New Design for Distrib-uted Systems: The Remote Memory Model," USENIX Summer Conference, 1990.
3) L. Ifrode et al., Compcon Spring '93, pages 538-547, 1993.
4) E. P. Markatos and G. Dramitinos, "Implementation of a Reliable Remote Memory Pager," USENIX 1996 Annual Technical Conference, 1996.
5) T. Newhall et al., Euro-Par 2003 Parallel Processing, vol. 2790, pages 1160-1169, 2004.
6) T. Newhall et al., IEEE Int. Conf. on Cluster Computing, pages 2-12, 2008.
7) M. D. Dahlin, et al., 1st USENIX OSDI, article no. 19, 1994.
8) G. M. Voelker et al., Joint Int. Conf. on Measurement and Modeling of Computer Systems, pages 33-43, 1998.
9) H. A. Jamrozik et al., ASPLOS-7, pages 258-267, 1996.
10) K. Li and K. Petersen, ISCA-18, pages 84-93, 1991.
11) M. Ekman and P. Stenstrom. "A Cost-Effective Main Memory Organization for Future Servers," IPDPS'05, 2005.
12) D. Ye et al., ICCD 2008, pages 272-279, 2008.
13) T. Kgil et al., 2006 Int. Conf. on Compilers, Architecture and Synthesis for Embedded Systems, pages 103-112, 2006.
14) D. Roberts et al., Communications of the ACM, vol. 52, no. 4, pages 98-106, 2009.
15) Y. J. Kim et al., IEEE Trans. on Consumer Electronics, vol. 53, issue 4, pages 1469-1476, 2007.
16) http://www.fusionio.com/PDFs/Data_Sheet_ioDrive_2.pdf
17) J. Suzuki et al., HOTI'06, pages 45-51, 2006.
18) H. Shimonishi et al., "A Congestion Control Algorithm for Data Center Area Communications," 2008 Int. CQR Workshop, 2008.
19) S. Floyd and T. Henderson, RFC 2582, IETF, 1999.
20) Io meter project: http://www.iometer.org/
21) pgbench: http://www.postgresql.org/docs/current/interactive/pgbench.html