社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－ARC－172 (15)
2007－HPC－109 (15)
2007／3／1

分散ページングによる大規模仮想メモリ空間

今井照之[†] 松葉浩也[††] 石川 裕[†,††]

今日のコモディティプロセッサは64 bitアドレス空間をサポートするにも関わらず，物理メモリの容量は限られている．本論文では，広大な仮想メモリ空間を実現する新しい分散リモートページングシステムDMPSを提案する．本システムは1つの計算ノードと複数のメモリサーバノードからなり，高速なネットワークで接続される．計算ノードはサーバと通信する仮想ブロックデバイスドライバを持ち，このデバイスがスワップ領域を提供する．本論文では，DMPSを設計し，Linux 2.6. 上にMyrinet Express (MX) インターフェースを用いて実装する．DMPSの性能を評価し，ローカルディスクを使用するシステムと比較し，ローカルディスクの2倍の性能を得た．

Large Virtual Memory Space using Distributed Paging

Teruyuki Imai,[†] Hiroya Matsuba[††] and Yutaka Ishikawa [†,††]

Though comodity processors support a 64 bit address space, the physical memory size is still limited. In this paper, we propose a new distributed remote paging system called DMPS to provide large virtual memory space. It consists of one computation node and multiple memory server node. Those nodes are connected by a high speed network. The computation node has a virtual block device driver to communicate with memory servers which provide a swap area. In this paper, DMPS is designed and implemented on Linux 2.6 with the Myrinet Express (MX) interface. The performance is evaluated and compared to a system using a local disk. The performance of the DMCP as swap is as twice as that of a local disk.

1. はじめに

今日，様々なアプリケーションの使用するメモリ容量は増加している．科学技術計算，大規模データベース，映像処理，遺伝子探索などは大容量のデータを扱うために大規模なメモリ空間を必要とする．また，科学技術計算やシミュレーションにおいては，より高い精度を求めるために可能な限り大きなメモリを使用する．

これに対応し，Intel x86アーキテクチャなどの今日のコモディティプロセッサも64bitアドレス空間をサポートする．しかしながら，1台のPCに搭載可能な物理メモリ容量は2GBから8GBと依然限られている．これより大きなメモリ空間を使用するために，共有メモリ型の並列コンピュータが使用される．共有メモリ型では従来のアプリケーションを改変なく使用できる．一方，今日普及しているPCクラスタでは，クラスタ全体のメモリ容量が十分である場合でも，これを利用するには，MPIなどの通信ライブラリを使用してアプリケーションを実装しなければならない．Linuxなど通常のPC上で通常のアプリケーションが物理メモリの容量を越える空間を使用するにはハードディスクをスワップとして用いるが，これは低速で性能低下を招く．例えば，DDR400 SDRAMの帯域幅は6.4GB/s, 10 Gb Ethernet, InfiniBand[1], Myrinet[2]など今日の高性能ネットワークの帯域幅が約1GB/sに対し，ハードディスクのアクセス速度は高々10^{-1} GB/sのオーダである．

本論文では，Distributed Memory Paging System (DMPS) という新しいリモートページングシステムを提案する．本システムの特色は以下の3点である．

- PCクラスタのOSとして標準的なLinuxを使用する．
- ホストのカーネルの改変は行わない．
- 特別なハードウェアに依存しない．

2. 設計と実装

2.1 概　　要

DMPSはクライアントプログラムDMCとサーバプログラムDMSからなる．DMCは仮想メモリシステムからのページング要求をDMSと通信して処理す

† 東京大学大学院情報理工学系研究科
Graduate School of InformationScience and Technology, The University of Tokyo
†† 東京大学情報基盤センター
Information Technology Center, The University of Tokyo

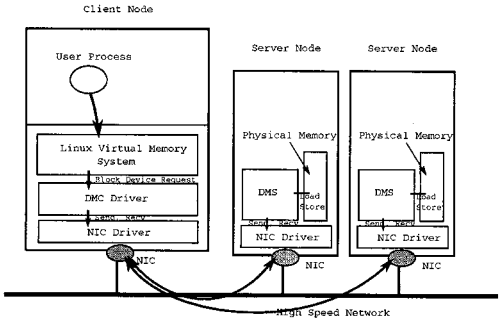

図 1 DMPS のアーキテクチャ

る．DMC は複数のメモリサーバの領域を 1 つの大きなディスクに見せる．DMPS のアーキテクチャを**図 1** に示す．

ページインは以下のように動作する．

(1) ブロックデバイス層はドライバへセクタの読み込み要求を発行する．
(2) DMC は読み込み要求を受けて NIC ドライバを用いてサーバノードに読み込み要求を送る．
(3) DMS はクライアントからの要求を受け取り，サーバノードのメモリ上にある，要求に対応するデータをクライアントノードに送る．
(4) DMC はサーバからのデータを受け取った後，Linux カーネルのブロックデバイス層に完了を通知する．

ページアウトは以下のように動作する．

(1) ブロックデバイス層はドライバへセクタへの書き込み要求を発行する．
(2) DMC は書き込み要求を受けて NIC ドライバを用いてサーバノードに書き込み要求と書き込むページデータを送る．
(3) DMS はクライアントからの要求とデータを受け取り，サーバノードのメモリ上の適切な場所にデータを格納する．
(4) DMS はクライアントに要求完了を応答する．
(5) DMC はサーバからの応答を受け，Linux カーネルのブロックデバイス層に完了を通知する．

2.2 メッセージサイズと帯域

1 回の通信当たりの転送データサイズは性能に影響する．一般にデータサイズが大きい程，帯域幅は大きい．Myrinet を用いた場合で ping-pong により測定した帯域幅を**図 2** に示す．なお，測定環境には本システムの評価環境 (**表 1**) を使用した．

Linux におけるページサイズは，多くのアーキテクチャにおいて 4KB または 8KB である．Linux のブロックデバイスの要求サイズは 2 の冪乗でページサイズ以下という制限があるため，ブロックデバイス層からの要求により単純に通信を行うとネットワークの性能を活用できない．このため，連続した要求を集約することが高性能を得るため重要である．

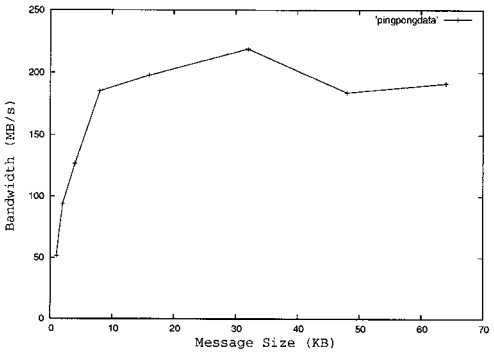

図 2 メッセージサイズと帯域幅の関係 (Myrinet XP)

2.3 複数サーバのサポート

1 つのサーバに搭載可能なメモリ容量もクライアント同様上限がある．そのため，大規模なメモリ空間を得るには，複数のサーバを使う事が必要である．複数サーバをサポートする場合，新たに考慮すべき点がある．まず，複数のサーバを使用する場合，RAID[11] のようにミラーリングやパリティにより，信頼性を得られる．サーバの使用容量の均等化も考えられる．また，クライアントが複数のネットワークインターフェースを持ち，異なるネットワークに接続されている場合，ストライピングにより，より高い性能を得る可能性がある．

2.4 DMC の構造

DMC の構造を**図 3** に示す．DMC は，Interface, Broker, Server Translator, Communicator 4 つのコンポーネントからなる．

Interface コンポーネントには，スワップとして使用するための Linux のブロックデバイスインターフェースを実装する．

Broker コンポーネントは Interface コンポーネントからの要求を Server Translator と Communicator の提供する API を用いて処理する．Server Translator は仮想セクタ番号とセクタ長のリストを受け取り，サーバのアドレス，サーバ上の実セクタ番号とセクタ長の組のリストに変換する．Communicator コンポーネントはサーバのアドレスと処理すべきセクタの番号を受け取り，ネットワークドライバを用いて通信を行う．

2.4.1 Interface

Interface は Linux ブロックデバイスインターフェースを持つコンポーネントである．このコンポーネント

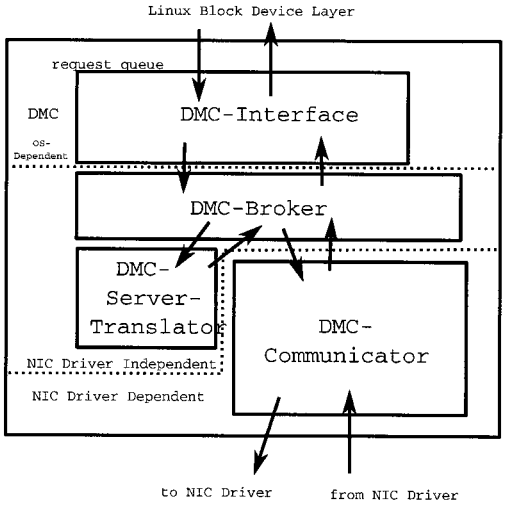

図 3 DMC デバイスドライバの構造

はカーネルのブロックデバイス層からの要求を前処理して Broker に渡す構造を構成する．例えば，物理連続なメモリセグメントをマージしたり，要求のサイズの検査を行う．このコンポーネントはネットワークとは独立している．また，このコンポーネントより下の層は本質的には OS の構造に依存しない．

2.4.2 Broker

Broker はブロックデバイスインターフェースと Communicator の仲立ちを行うコンポーネントである．Interface からは，スキャッタギャザリスト形式になった，バッファと仮想セクタ番号のリストを受け取る．このコンポーネントは Server Translator によって仮想セクタをサーバアドレスとサーバの実セクタに変換し，その結果を元に，Communicator を使ってサーバへ要求を発行し，さらに Communicator を使って要求の完了を確認する．このコンポーネントは Interface コンポーネントのためのインターフェースの他，他のカーネルモジュールからの利用の便宜のため，物理連続なバッファとセクタ番号，セクタ数を渡す事で通信を行うインターフェースも持つ．

2.4.3 Server Translator

Server Translator は，DMPS に参加するサーバノードのリストを管理し，サーバの提供する空間の使用に関するポリシを実装するコンポーネントである．このコンポーネントは，各サーバのセクタ数とアドレスを保持する．要求を受け取り，サーバのアドレスとサーバ上の実際のセクタ番号を引き，その結果を書き込む．「アドレス」はネットワークドライバに依存するが，このアドレスは単純に保持し，要求を渡された時に，要求のアドレスフィールドに複写するだけで，このアドレスを用いた通信操作は行わない．従って通信ドライバには依存しない．

2.4.4 Communicator

Communicator は，MX API のような通信 API をラップしている．このコンポーネントが読み込みと書き込みの要求開始と通信の完了を確認するインターフェースを提供する．要求の発行は，メモリセグメントのリスト，サーバのアドレス，サーバ上における実セクタ番号の組を渡す事で行う．

2.5 DMC における複数サーバサポート

DMC は，1 つのサーバノードの物理メモリより大きな空間の使用を可能にするために，複数のサーバを使用可能としている．我々の手法では，システムは 1 つのブロックデバイスを持つ．Server Translator コンポーネントは複数のサーバ空間のそれぞれのセクタに統一的な仮想セクタ番号を与え，サーバにより提供される空間を 1 つの大きな空間に見せる．他の手法としては，複数のブロックデバイスをスワップデバイスとして使用する方法がある．この手法では，サーバの空間の使われ方は，Linux の仮想メモリシステムに依存する．よって，メモリクライアントはサーバの使われ方を制御できない．

2.6 実　　装

DMPS を Linux 2.6.18.5 上に，MX API 1.1.6 を使用して実装する．

2.6.1 DMC の実装

DMC は Linux のブロックデバイスドライバモジュールとして実装する．クライアントホストのカーネルは変更しない．

Interface はブロックデバイスへの要求を処理するストラテジルーチンを実装する．また，ユーザから DMPS を操作するための `ioctl` のルーチンも提供する．セクタサイズは多くのアーキテクチャのページサイズと同じく 4KB，1 リクエスト当たりの最大セクタ数を 128KB とした．これは大きい程転送効率は高くなるが，同じ大きさの物理連続なバッファを Communicator 内に割り当てる必要があるトレードオフがある．

Broker コンポーネントの実装は単純である．Server Translator によって仮想セクタからサーバのアドレスと実セクタのリストに変換した後，Communicator API により要求を発行し，全ての要求が完了するまで確認関数の呼出を繰り返す．

本論文における Server Translator の実装は以下の通りである．仮想セクタから実サーバとセクタ番号へのマッピングは，サーバのセクタに対し，前から順に仮想セクタを対応させている．すなわち，仮想セクタ番

号 s_v はサーバ n のセクタ s_r に対応するとすれば，

$$s_v = s_r + \sum_{i=0}^{n-1} S_i \quad (1)$$

となる．但し，S_i はサーバ i の総セクタ数である．本実装の Server Translator には，ストライピングを実装していない．クライアントが 1 つのネットワークインターフェースしか持たない場合，ストライピングによる性能向上は期待できないためである．また，ミラーリングやパリティロギングも行わない．今回の実装では，DMPS に参加する最大サーバ数を 16 に制限した．要求に関して，開始セクタと終了セクタを逐次的に探索する．サーバ数が多くないため，これは問題にならない．

Communicator は Myricom 社の Myrinet[2] ネットワーク用の MX API を用いて実装した．この Communicator は，128KB の物理連続なバッファを 4 つ持つ．それぞれ読み込みと書き込み用に 2 つずつである．現在の Myrinet NIC とドライバの制限により，RDMA 転送を行うデータサイズにおいて，NIC レベルのスキャッタギャザが不可能という制限に対応し，かつ転送効率を高めるため，読み書きは以下のように行う．サイズが小さい場合は MX の持つスキャッタギャザ機能を用いて行う．サイズが大きく RDMA 通信を行う閾値を越えたものに関しては，転送バッファのセグメントが 1 つの場合は直接 RDMA を行い，複数ある場合は Communicator 内部のバッファを用いて Communicator 自身がスキャッタギャザ処理を行い，転送は 1 回の RDMA により行う．

2.6.2 DMS の実装

DMS のプロトタイプは単純なユーザプログラムとして実装した．クライアントからの読み込み，書き込み，制御のコマンドと書き込みにおけるデータの受信のための，規定個数の受信要求を予め発行しておく．これにより，クライアントからのメッセージが Unexpected な状態になる事を回避する．

2.6.3 MX ドライバの改変

MX ドライバをスワップ機構の実装に使用可能とするために，以下ののような変更を加えた．

MX ドライバをスワップ要求の処理に使用する時にデッドロックを回避するため，メモリ割り当てのフラグを `GFP_ATOMIC` に変更した．このフラグは，高優先度によるメモリ割り当てを要求し，スリープしてはならないという事を意味する．通常のフラグ `GFP_KERNEL` により割り当て要求を行うと，メモリ割り当て中のスワップ処理においてスリープし，デッドロックする可能性がある．

表 1　評価用計算機の諸元

CPU	Intel Xeon Processor 2.80GHz × 2
メモリ	2GB
PCI	PCI-X (64bit 133MHz)
ネットワーク	Myrinet XP Scalable Cluster Interconnect
SCSI コントローラ	Adaptec AIC-7901A U320
ハードディスク	73GB 15000RPM Seagate ST373453LC
OS	Linux 2.6.18.5

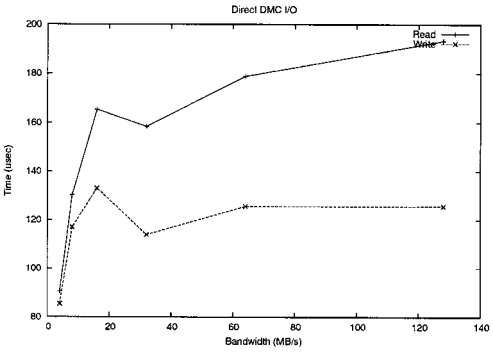

図 4　Direct DMC I/O による帯域幅

3. 評　　価

DMPS を 1 つのクライアントノードと 1 つまたは 2 つのサーバノードを用いて評価する．評価環境を表 1 に示す．クライアントノードは，起動時のオプション `mem=128M` によりカーネルから使用可能なメモリ容量を 128MB に制限している．カーネルパラメタ *page-cluster* 変数を 1 に設定した．これはページインする時 1 度に読み取るページ数を設定するパラメタで，デフォルト値は 3 である．仮想メモリシステムは 1 度に $2^{page\text{-}cluster}$ ページ読み取る．この値を変更する事で，ディスク，DMPS いずれをスワップにした場合も性能向上が見られた．

3.1 Direct DMC I/O

テストプログラムは 128KB の物理連続なバッファを `alloc_pages` により割り当て，ブロードカのインターフェースを用いて読み込み，書き込みアクセスを行う．読み書きによる所要時間を `do_gettimeofday` 関数により測定する．このテストでは 32MB のバッファを用意したサーバ 1 つにより評価する．

図 4 に得られた帯域幅を示す．読み込みにおける帯域幅は 128KB 転送時に最大で，帯域幅は 194MB/s, 書き込みにおける帯域幅も 128KB 転送時に最大で，帯域幅は 126MB/s となる．

3.2 IOzone

IOzone[10] はファイルシステムのベンチマークである．このベンチマークはシーケンシャルな書き込みと

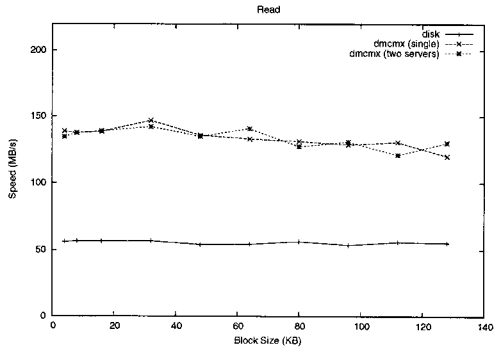

図 5　IOzone 読み込みブロックサイズと速度

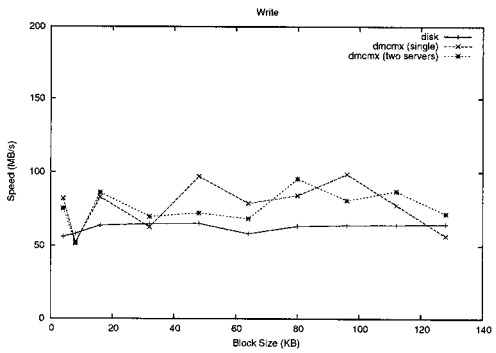

図 6　IOzone 書き込みブロックサイズと速度

読み込み，ランダムな書き込みと読み込み，そして逆順の書き込みと読み込みを行い，所要時間をブロックサイズ (1 回のアクセスで読み書きするサイズ) を変化させながら測定する．

本実験では，実験用のファイルサイズをメモリサイズの 2 倍である 256MB として，書き込みと読み込みの速度測定を行う．DMPS の性能評価には 320MB のメモリサーバを使用し，DMC 上に ext3 ファイルシステムを作成する．この時，320MB 割り当てた DMS を 1 つ使用する場合と，160MB を割り当てた 2 つの DMS を使用する場合の両方で測定する．これらの結果をローカルディスクを使用した場合と比較する．

読み込みの測定結果を**図 5** に，書き込みの測定結果を**図 6** に示す．　DMPS の読み込み速度は 120MB/s~147MB/s と，ディスクの 2 倍以上の速度である．一方，書き込み速度は 51MB/s~96MB/s である．これはディスクの場合との差があまりない．いずれの場合も，サーバが 1 台の場合と 2 台の場合で性能に差は見られない．

3.3 Sequential

このテストプログラムは，指定された容量を malloc し，その領域に，前から順番に 32bit 整数を 1 つずつ書き込み，その後，前から順番に 1 つずつ読み込む．このテストのために 1 GB のスワップをハードディスクのパーティションを使用した場合と DMPS を使用した場合で比較する．DMPS のテストには，1GB 割り当てた DMS を 1 つ使用する．割り当てるメモリ容量を 64MB から 512MB まで変化させて測定する．

結果を**図 7** に示す．

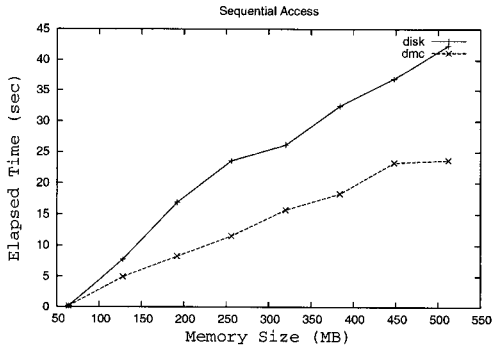

図 7　シーケンシャルアクセスの所要時間

4. 関連研究

リモートホストのメモリをネットワークを介して利用する研究は古くからある．まず，仮想メモリシステムを置き換える事による実現としては Iftode らによる研究[6] や GMS[4] がある．これらのシステムでは，物理メモリ管理機構，ページフォルトハンドラ，ページ置換機構を実装している．また，類似の手法として，ソフトウェア分散共有メモリシステムを用いた Cashmere-VLM[3] がある．この手法では，Linux の場合，カーネル本体の改変なしには実現できない．

スワップによるアプローチとしては，以下の研究がある．Ethernet LAN 上でのシステムとしては Remote Memory Pager (RMP)[9] や Network RamDisk (NRD)[5] がある．これらは TCP/IP を用いている．これらの研究では信頼性に焦点を置いており，パリティロギングにより信頼性を実現している．

HPC 用のネットワークを使用した研究としては以下のものがある．HPBD[7] は InfiniBand を使用したスワップ用の仮想ブロックデバイスである．この研究では，InfiniBand で RDMA を効率的に使用する事を中心に考えられている．NUZURA[13] は FPGA を搭載した 10 Gb Ethernet NIC UZURA[12] を用いたスワップデバイスである．UZURA 上の FPGA を用いて独自の RDMA データ転送を実現し，データは独自の Ethernet フレームにより転送する．

Network Block Device (NBD)[8] は Linux カーネル本体に付属したリモートホストのブロックデバイスを使用可能にする仮想ブロックデバイスドライバである．サーバ側のホストで RAM ディスクと NBD サーバを組み合わせる事で，TCP/IP ネットワークを用いたリモートページングシステムを構築できる．NBD をスワップとして用いると，スワップ中の TCP/IP 層でのメモリ割り当てによりデッドロックを起こす場合がある．

5. おわりに

本研究では，大規模なメモリ空間を使用可能にするために，新しいリモートページングシステムを提案した．このシステム DMPS を設計し，Linux 2.6 上に Myrinet Express ドライバを使用して実装した．このシステムを Xeon PC クラスタ上で評価し，読み込み性能についてはハードディスクを使うシステムの 2 倍の性能を得たが，書き込み性能はハードディスクとの差が見られず，スワップとして利用した場合の性能も改善がみられなかった．

今後の研究課題としては，性能改善と，実アプリケーションによる評価，複数のネットワークで接続したメモリサーバに対するメモリサーバ使用ポリシの検討などが挙げられる．また，DMPS をスワップとした場合，ディスクをスワップとした場合のいずれも，*page-cluster* 変数を大きくした時に性能劣化が観察された．この原因の考察も必要である．

謝　辞

本研究の一部は、科学研究費補助金基盤研究 (B) 課題番号 18300006「次世代 PC クラスタを活用する超大規模仮想メモリ空間支援システムの研究」による。

参考文献

1) InfiniBand Trade Association. `http://www.infinibandta.org/`.
2) Myrinet. http://www.myri.com/.
3) Sandhya Dwarkadas, Nikolaos Hardavellas, and Leonidas Kontothanassis. Cashmere-VLM: Remote Memory Paging for Software Distributed Shared Memory. In *Proceedings, 13th International Parallel Processing Symposium and 10th Symposium on Parallel and Distributed Processing*, 1999.
4) Michael J. Feeley, William E. Morgan, Frederic H. Pighin, Anna R. Karlin, Henry M. Levy, and Chandramohan A. Thekkath. Implementing Global Memory Management in a Workstation Cluster. In *Proceedings of the fifteenth ACM symposium on Operating systems principles*, 1995.
5) Michail D. Flouris and Evangelos P. Markatos. The Network RamDisk: Using Remote Memory on Heterogeneous NOWs. *Journal of Cluster Computing*, 1999.
6) Liviu Iftode, Kai Li, and Karin Petersen. Memory Server for Multicomputers. In *Proceedings of the 38th IEEE International Computer Conference (COMPCON Spring '93)*, 1993.
7) Shuang Liang, Ranjit Noronha, and Dhabaleswar K. Panda. Swapping Remote Memory over Infiniband: An Approach using a High Performance Network Block Device. In *IEEE International Conference on Cluster Computing (Cluster 2005)*, 2005.
8) Pavel Machek. Network block device. `http://nbd.sourceforge.net`, 1999.
9) Evangelos P. Markatos and George Dramitinos. Implementation of a Reliable Remote Memory Pager. In *Proceedings of the 1996 Usenix Technical Conference*, 1996.
10) William D. Norcott. Iozone filesystem benchmark. `http://www.iozone.org/`.
11) David A. Patterson, Garth Gibson, and Randy H Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID). In *Proceedings of the 1988 ACM SIGMOD international conference on Management of data*, 1988.
12) 中島耕太, 佐藤充, 後藤正徳, 住元真司, 久門耕一, 石川裕. 配列転置データ転送を高速化する 10 Gb Ethernet インターフェースカードの設計. 先進的計算基盤システムシンポジウム SACSIS2006, 2006.
13) 後藤正徳, 佐藤充, 中島耕太, 久門耕一. 10Gb Ethernet 上の RDMA を用いた遠隔スワップメモリの実装. 電子情報通信学会技術研究報告 Vol.106 No. 287, 2006.