

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 commodity 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 DMPS 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、InfiniBand¹⁾、Myrinet²⁾ 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 のオーダーである。

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。本システムの特徴は以下の 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 Information Science 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

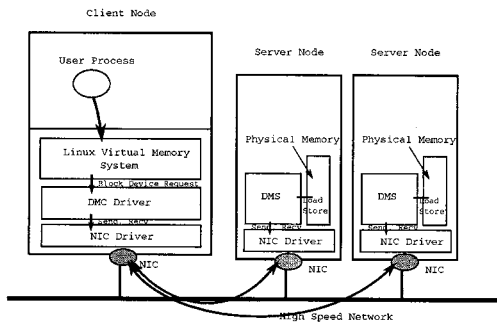


図 1 DMPS のアーキテクチャ

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

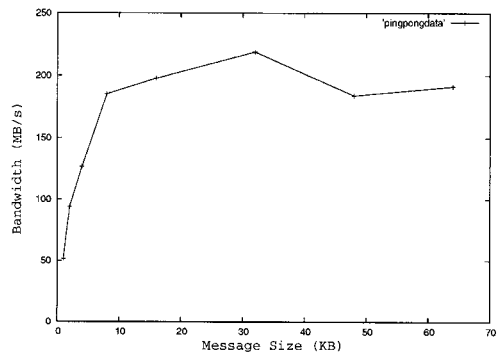


図 2 メッセージサイズと帯域幅の関係(Myrinet XP)

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。

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^[1]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

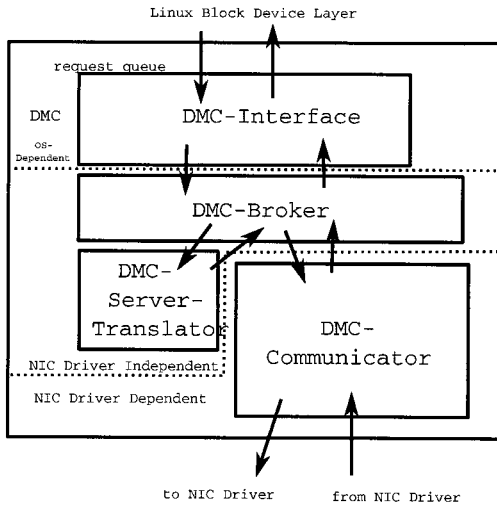


図3 DMC デバイスドライバの構造

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、スキヤッタギ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ー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イ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 Myrinet²⁾ 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 レベルのスキヤッタギヤ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 の持つスキヤッタギヤ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 自身がスキヤッタギヤザ処理を行い、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

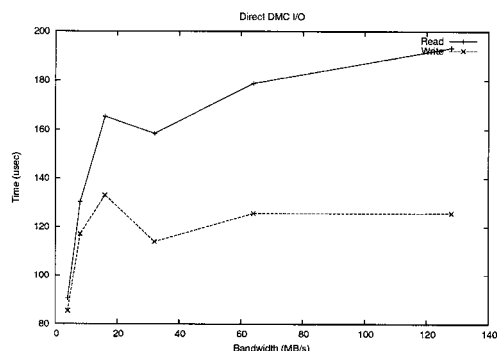


図 4 Direct DMC I/O による帯域幅

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 `2page-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、ブローカのインターフェイ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

IOzone¹⁰⁾ 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

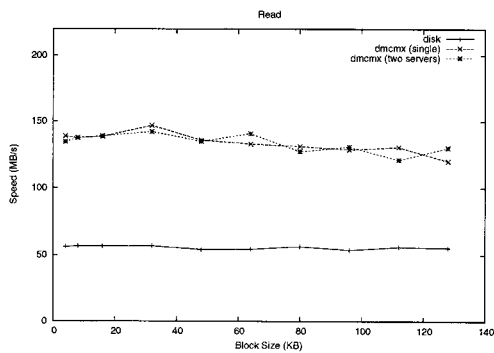


図5 IOzone 読み込みブロックサイズと速度

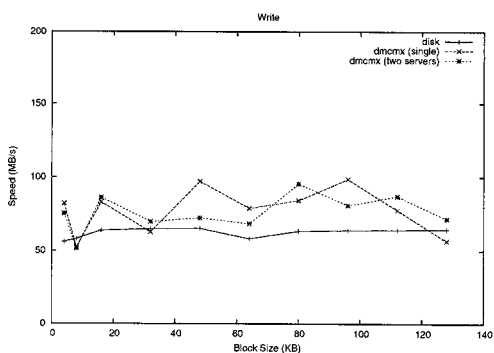


図6 IOzone 書き込みブロックサイズと速度

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

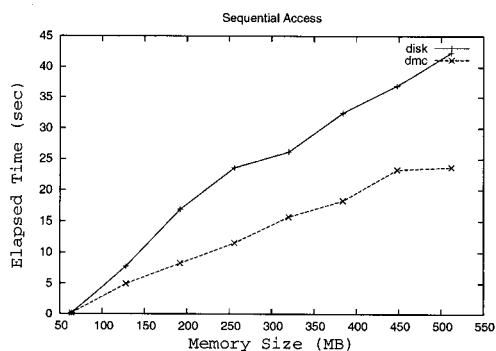


図7 シーケンシャルアクセスの所要時間

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1GB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。割り当てたメモリ容量を64MBから512MBまで変化させて測定する。結果を図7に示す。

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GMS⁴⁾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Cashmare-VLM³⁾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、バリエーション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。HPBD⁷⁾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。NUZURA¹³⁾はFPGAを搭載した10 Gb Ethernet NIC UZURA¹²⁾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ー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.