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

2007－HPC－111 (21)
2007／8／2

# 大規模メモリ空間の利用を支援する遠隔スワップメモリシステム

北村裕太[†] 松葉浩也[††] 石川 裕[†]

64ビットアーキテクチャPCにおいて，一台のPCが持つ物理メモリ容量以上の仮想メモリ空間を利用できるようにするための遠隔スワップメモリシステムをLinuxカーネル上に設計する．予備実験として，ネットワークブロックデバイスを用いたLinuxのスワップ機構を評価し問題点を示す．次に，Linuxのスワップ機構とは異なる新しい遠隔スワップメモリシステムを設計する．本システムでは，ネットワークブロックデバイスが効率良くデータ転送を送れるように，連続物理ページ単位で領域を管理する．ユーザが定義できるページ入れ換えアルゴリズム機構を提供することにより，アプリケーションのメモリアクセスパターンによる投機的ページ入れ換えを可能とする．

# A Remote Memory Swapping System Facilitating Usage of Large Memory Space

Yuta Kitamura,[†] Hiroya Matsuba[††] and Yutaka Ishikawa[†]

A new remote swap system in the Linux kernel, running on a 64 bit PC, is designed in order to provide virtual address memory whose size is larger than the physical main memory. As an experiment, the Linux swap mechanism is evaluated using a network block device. We, then, propose a new remote memory swapping system which differs the original Linux swap system. In the system proposed, the contiguous physical pages of a user process are managed so that the network device may send and receive physical pages efficiently. A new system call, defining page replace algorithm, has been introduced to realize speculative page replacement based on the application memory access pattern.

## 1. はじめに

近年，大規模なメモリ空間を有効に活用できるプラットフォームの必要性は高まりつつある．例えば，遺伝子の探索やマルチメディア処理，大規模データベースなど，大容量のデータを扱うアプリケーションは大きなメモリ空間が必要となる．また，シミュレーションや数値計算などの科学技術計算では，計算の規模を高めるため，あるいは計算結果の精度を向上させるために可能な限り大きなメモリを利用できることが望ましい．

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利用されている．共有メモリ型マシンを用いると，従来のアプリケーションをほぼ改変することなく，より大きなメモリ空間に対応させることができる．しかし，大規模共有メモリ型並列コンピュータは高価であり，また，PCの計算性能が飛躍的に向上してきたことから，多数のPCを高速なインターコネクトで接続するPCクラスタが広く利用されるようになった．PCクラスタは分散メモリ型並列コンピュータである．PCクラスタの全メモリを利用するためには，MPIなどの通信ライブラリを用いてアプリケーションを実装し直す必要がある．

大容量メモリを要求するアプリケーションの重要性の高まりを受け，PC向け64ビットマイクロプロセッサが広く普及している．例えば，IntelやAMDの64ビットアーキテクチャでは，ともに少なくとも48ビット以上の仮想アドレス空間をサポートしており，これは理論上，256TB（テラバイト）の仮想アドレス空間を扱うことができるということを意味する．しかし，現代のPCに搭載可能なメモリは高々数GBから数十GBであるため，それより大きなメモリ空間を利用しようとすると，アクセス速度がメモリに比べて非常に遅いハードディスクにメモリの内容を退避するメモリスワッピングを行う必要がある．一旦メモリスワッピングが発生すると計算性能は大きく落ちてしまうため，実メモリより遥かに大きなメモリ空間を要求するアプリケーションをPCで実行するのは現実的ではない．

† 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

本論文では，アプリケーションを改変せずに大規模メモリ空間を利用できるようにするために，ハードディスクよりスループット性能が高い高速インターコネクトの利用を仮定する．OS として Linux を用い，ホスト同士を Myrinet Express[7] で接続した PC クラスタ上で，遠隔ホストのメモリをスワップデバイスとして用いる遠隔スワップメモリシステムを設計する．

大規模メモリ容量を提供するシステムを構築する方法には，大きく，i) OS のスワップ機能を用いる遠隔スワップメモリシステムを実現する，ii) ソフトウェア分散メモリシステムを実現する，という 2 つの方法がある．本稿では，Linux スワップ機構を用いる遠隔スワップメモリシステムについて予備実験を行い，問題点を考察する．その結果，Linux スワップ機能をネットワーク経由で使う場合に性能面での問題があることを指摘する．そこで，本論文では，character デバイスによるメモリマップ機能およびページフォルト機能を使用した遠隔スワップ用デバイスを設計する．

## 2. Linux のスワップ機構

Linux の仮想メモリレイヤでは，基本的にページ単位（多くの場合 4KB の大きさを持つ）で処理が行われる．スワッピングも仮想メモリ機構の一部であるので，スワップアウト・スワップインも基本的にページを単位として行われる．

物理メモリ上にあるページが不足すると，現在メモリ上にあるページをスワップデバイスに書き出し，メモリ上にあるページを解放するスワップアウトが行われる．スワップデバイス上に存在するページを利用するときには，一旦メモリ上にページの内容を読み込み，スワップデバイス上のページを解放するスワップインが行われる．スワップデバイスとしては，ブロックデバイスか，ブロックデバイス上のファイルシステムにあるファイルを指定することができる．

スワップアウト処理は以下のような場合に起動される．

- バッファやページの割り当てに失敗したとき．
- 利用可能なページの数が一定の閾値を下回ったとき（カーネルスレッド kswapd により起動される）．

スワップアウトの対象となるメモリ上のページは，一般に擬似的な LRU アルゴリズムを用いて決定される．Linux では，アクティブなページのリストと非アクティブなページのリストを管理している．ある一定の期間に連続してアクセスがあるとページはアクティブなリストに移され，逆にしばらくアクセスがないとページは非アクティブなリストに移される．スワップアウトが行われるときには，非アクティブなリストからページを取り出し，スワップデバイスに移動される．

スワップデバイス上のどこに保存するかはスワップアウトのたびにそれぞれのページについて計算され，連続したスワップアウト要求では，スワップデバイス上でなるべく連続した位置に保存するようにする．これには，ハードディスクをスワップデバイスとした場合に，シーク時間を削減することと，関連したページをなるべく近くに集める（クラスタ化する）ためという 2 つの理由がある．

スワップイン処理は，ユーザプロセスが既にスワップアウトされたページにアクセスしようとして，ページフォルトが発生したときにページフォルトハンドラから起動される．スワップインするときは，スワップデバイス上で取得するべきページ以降の一定ページ数を連続してスワップインしようとする（`swapin_readahead` 関数）．具体的には，カーネル内の変数 *page-cluster* を用いて，$2^{page\text{-}cluster}$ ページだけの先読みが行われる．この先読みも，シーク時間の削減とページのクラスタ化を目指して実装されている．

スワップアウト・スワップイン要求は，ページ単位のブロックデバイスへの要求に変換され，ブロックデバイスへのアクセスを抽象化する汎用ブロックレイヤを通過し，最終的にデバイスドライバが要求を処理する．汎用ブロックレイヤは，ブロックデバイスへの要求を一定時間ため込み，可能ならば要求のマージを行い，デバイスドライバへ処理を引き継ぐ役割を担っている．

デバイスドライバが転送を終えると，不要になったメモリ上のページ（スワップアウトの場合）あるいはスワップデバイス上のページスロット（スワップインの場合）が解放される．

以上が Linux のスワップ機構の処理の大まかな流れである．

**表 1** 評価用計算機の仕様

| | |
|---|---|
| CPU | Dual Core AMD Opteron 270 ×2 |
| メモリ | 2GB |
| ディスク | HGST Deskstar T7K250 (Ultra DMA/133) |
| PCI | PCI-X 133MHz |
| ネットワーク | Myricom Myrinet M3F-PCIXD-2 |
| ドライバ | MX-2G 1.1.8 |
| OS | CentOS 5, Linux 2.6.18.5 |

## 3. 予備実験

### 3.1 実験環境

予備実験に用いた計算機の仕様を **表 1** に示す．使用したネットワークは 2Gbps Myrinet である．予備実験では，この計算機を 2 台接続して実験を行った．スワッピングの性能を計測する際は，実行ホスト側の計算機のカーネルのコマンドラインオプションで `mem=128M` を指定することにより，利用できるメモリの大きさを 128MB に制限して実験を行った．

遠隔スワッピングを行うために，MX 通信ライブラ

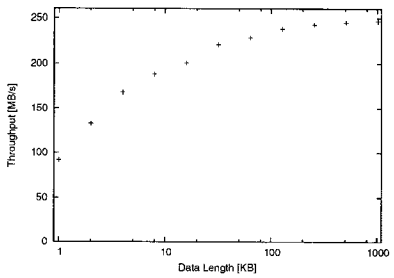


**図 1** 送信単位サイズとスループット性能

リ の API を用いて，遠隔ホストをメモリサーバとして利用する遠隔ページングシステム DMPS[11] を利用する．DPMS では，MX 通信ライブラリの API を用いてブロックデバイスを作成する．遠隔ホスト側で定義されるブロックデバイスへの I/O 要求は，MX 通信ライブラリを使用してメモリサーバに送られる．

メモリサーバは，遠隔ホストからの I/O 要求を受信するとメモリ内に保持してあるデータを使って処理を行う．メモリサーバが提供するメモリ容量は物理メモリ量を越えない値とする．これにより，メモリサーバ側でスワップは生じない．

ネットワーク経由によるブロックデバイスを提供することにより，スワップ機構を変更することなく，本ブロックデバイスをスワップデバイスとして使用できる．これにより，ハードディスクより高速なブロックデバイスが実現される．

DMPS のブロックデバイスをスワップデバイスとして登録し，利用可能なメモリ（128MB）より大きなメモリを要求するアプリケーションの実行時間を計測した．

各実験の開始時には，一旦 `swapoff` してから `swapon` することにより，スワップアウトされているページを一旦メモリに読み戻した．

### 3.2 ネットワーク性能

実験機 2 台間で MX を通じて通信を行ったとき，送信単位サイズによってスループットがどのように変化するかを **図 1** に示す．MX-2G 付属のソフトウェア `mx_pingpong` をベンチマークプログラムとして用いた．

図 1 によると，送信単位サイズが概ね 1MB を上回るとほぼ最大のスループットを出すことができるが，ページサイズ（今回の場合は 4KB）ずつ送信すると最大 7 割程度の性能しか出ないということがわかる．

### 3.3 ベンチマーク性能

#### 3.3.1 逐次アクセスに要する実行時間

単純なパターンでメモリをアクセスするプログラムの実行時間を計測した．テストプログラムは，利用可能なメモリの 4 倍の 512MB の大きさのメモリを `malloc` で確保し，一旦メモリ領域の先頭から最後までをアクセスする．その後，逐次的にメモリの先頭から最後までをページサイズおきに順番にアクセスし，時間を測定した．

測定は，kswapd を動かす・動かさない，スワップイン時の先読みページ数を 1 にする場合と 8 にする場合の組み合わせで行った．結果を**表 2** に示す．表中の「オンメモリ」とは，利用できるメモリの大きさを制限せず，テストプログラム全体が物理メモリに格納されている場合を意味する．minor fault 数および major fault 数とは，メモリ領域初期化後のテストプログラム実行中のページフォルト回数を示している．major fault 数とは，ページフォルトが生じて，ページをスワップインするためにテストプログラムが停止した回数であり，minor fault 数とは，ページフォルトが生じてもプロセスのページが見つかりすぐに復帰した回数のことである．

minor fault 数と major fault 数の合計は，いずれの場合もアクセスするメモリ領域のページ数（512MB/4KB = 131072 ページ）にほぼ等しかった．スワップインの先読みページ数が 1 の場合，スワップ領域からのページの読み込みは 1 ページずつ行われるため，major fault 回数はアクセスするメモリ領域全体のページ数とほぼ等しくなる．先読みページ数を 8 にすると，major fault 時に先読みが行われるため，先読みが行われた部分のページフォルトは minor fault になり，major fault 回数が減少する．

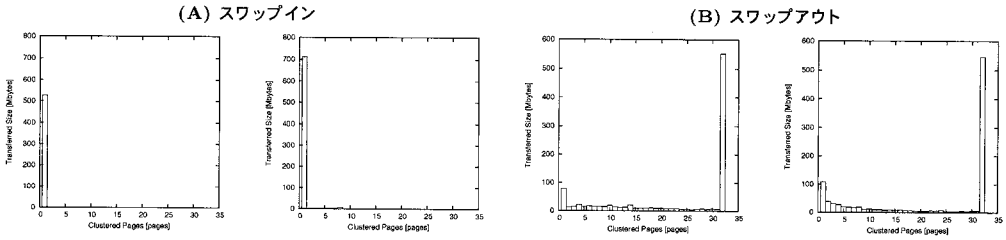
表 2 によると，kswapd を無効にするとハードディスクをスワップデバイスと用いた場合には 3 割ほど実行速度が速くなったのに対し，遠隔スワップメモリを用いた場合には逆に実行時間が倍近くまで増加した．このような結果になる理由は現在のところわかっておらず，さらなる調査が必要である．

**図 2** は，同じテストプログラムを kswapd が有効なオリジナルのカーネル上で実行したときのメモリサーバ側で受信されたメモリの要求サイズのヒストグラムである．kswapd が無効の場合でも結果は 図 2 とほぼ同様の結果であった．スワップアウト時は，送信データの大部分において 32 ページ分の要求がまとめて送信されているが，スワップイン時はほとんどが 1 ページずつの要求となっている．また，ページクラスタ数を大きくすると，スワップイン時に本来必要なデータ量の 512MB より大きい転送が発生している．これは，スワップインの先読み時にはスワップ領域で連続したページを先読みするため，テストプログラム以外のプロセスのページがスワップインされたためであると考えられる．

表 2 によると，ページクラスタ数を増やすことによりプログラムの実行時間が遅くなっている．特にシークに時間を要するハードディスクの場合は，スワップインの先読みページ数を増やして major fault が減少させることで実行時間の高速化に有利に働くと想像さ

表 2　逐次アクセスに対する性能

| 環境 | | ページクラスタ数 | 時間 (秒) | minor fault 数 | major fault 数 |
|---|---|---|---|---|---|
| オンメモリ | | – | 0.136 | | |
| kswapd 有効 | ディスクスワップ | 1 | 154.8 | 7 | 131016 |
| | ディスクスワップ | 8 | 164.7 | 108840 | 22142 |
| | ネットワークスワップ | 1 | 12.6 | 4 | 131075 |
| | ネットワークスワップ | 8 | 18.6 | 107571 | 23509 |
| kswapd 無効 | ディスクスワップ | 1 | 108.7 | 5 | 131074 |
| | ディスクスワップ | 8 | 109.5 | 111526 | 19551 |
| | ネットワークスワップ | 1 | 23.1 | 4 | 131075 |
| | ネットワークスワップ | 8 | 29.4 | 108306 | 22773 |




(a) ページクラスタ数 1　(b) ページクラスタ数 8　(a) ページクラスタ数 1　(b) ページクラスタ数 8

図 2　メモリ要求サイズのヒストグラム

れるが，major fault 減少による高速化の効果より無駄なスワップイン I/O の増加による消費時間の増加の方が影響が大きかったものと考えられる．

### 3.4 考　　察

#### 3.4.1 ページのスキャッタ・ギャザー

図 2 から分かる通り，スワップインのほとんどのメモリ要求サイズは 4KB(1 ページ分) であることが分かる．一方，図 1 によれば，4KB のデータ転送性能は，最大性能の 7 割程度である．すなわち，ページ単位で送信するのではなく，ある程度ページをまとめて送信しなければ高スループットが達成できないということがわかる．Myrinet-2000 における最大スループットを出そうとすると，1MB の単位での送信が必要となる．ページサイズ 4KB の環境では 256 ページ程度まとめて送受信することが必要となる．

第 2 節で Linux のスワップ機構を紹介した通り，Linux のスワップ機構のクラスタ化機能を使用すると，ページが連続されて利用される可能性がある．本機構はディスク上の配置を考慮しているだけで，物理ページが連続して割り当てられることを考慮していない．実際，Linux 2.6.20 のソースプログラムを調べると以下のことが分かる．

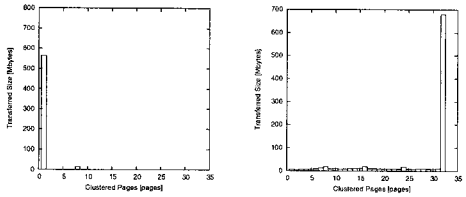
`swapin_readahead` 関数において，連続したページをスワップインしようとするが，物理ページは 1 ページ単位で取得する．Linux では，連続して 1 ページの物理ページを取得しても，物理的に連続した領域が確保されない．取得した物理ページにスワップアウトされている仮想アドレスページの内容を読み込むために，`swapin_readahead` 関数は汎用ブロックデバイス層を通じて非同期 read 要求を出す．

汎用ブロックデバイスでは，非同期 read 要求の待ち行列を保持している．非同期 read 要求列における読み込み先物理ページが連続している場合，それら要求列を一つにまとめる．`swapin_readahead` 関数が 1 ページ単位で物理ページを取得しているため，連続した仮想アドレス領域を一括してスワップインするときでも物理ページは非連続になっているため，ブロックデバイス側はスワップイン要求を一括することができなくなる．

Linux の物理メモリ管理は Buddy System を採用していて，2 の冪乗単位で連続物理ページを確保することが可能である．`swapin_readahead` 関数において，物理ページを 1 ページ単位で取得するかわりに連続物理ページを確保した上で，ブロックデバイスに対して非同期 read 要求を出せば，ブロックデバイス側でスワップイン要求を一括処理可能となる．

そこで，`swapin_readahead` 関数を修正して実験した．**図 3** にメモリサーバ側で受信されたメモリの要求サイズのヒストグラムを示す（kswapd が有効で，ページクラスタ数 8 の場合）．スワップアウトの場合はかなり集約されるようになったが，スワップインはあまり改善しなかった．今後，さらに調査する必要があるが，少なくとも，現状の Linux スワップ機構では，常にスワップイン要求を一括処理するのは難しいといえる．

現状の Linux を修正することなく高効率でページ




(A) スワップイン　　(B) スワップアウト

**図 3**　物理連続ページを割り当てた場合

データの送受信を行うためには，ブロックデバイスのドライバ側でスキャッタ・ギャザー処理を行う方法がある．

### 3.4.2 kswapd

kswapd は，システムで利用可能なページが一定の閾値を下回ったときに動作し，自律的にページキャッシュの破棄・ページのスワップアウトを行い，利用可能なページを確保するカーネルスレッドである．上記の逐次アクセスのプログラムを連続して何回か実行したところ，kswapd を動作させない方が安定した結果が得られることがわかった．これは，kswapd が独立して動作するため，毎回の動作パスが安定せず，近い将来に必要なページをスワップアウトしてしまうケースが発生したためであると考えられる．

kswapd を止めることでスワップの性能は向上することはあるが，一般に kswapd を動作させないことは良い考えではない．メモリの割り当ては割り込みハンドラや例外ハンドラ内で発生する可能性があるため，I/O を開始できないコンテキストからメモリ割り当てルーチンが呼ばれ，メモリ不足によりメモリ割り当てに失敗した場合，OOM Killer（Out of Memory Killer）によってプロセスが強制的に kill される可能性がある．実際これは稀なケースであるが，OOM Killer の動作は好ましくないため，kswapd を無効にしてメモリが常に不足する状態を維持するのは危険である．

### 3.4.3 スワップ領域の扱い

Linux がスワップ領域にページを配置する際，スワップアウト要求の順になるべく連続するように配置される．スワップイン時には，スワップ領域の当該領域は解放される．このため，スワップアウト対象のページがスワップインからデータが変更されていなくてもスワップ領域にスワップアウトする必要がある．本機構により，実行中の任意の時点でのワーキングセットをとっても，連続したスワップ領域にはプロセスのメモリ空間上で連続したページが保存される可能性が高い．これは，ランダムアクセスに時間がかかるディスクのようなデバイスに対しては転送時間の短縮につながる．一方，ランダムアクセス時間が一定のメモリサーバ方式の場合は，ネットワークトラフィックの軽減が重要となる．

## 4. リモートスワップ機構設計に向けて

予備実験および考察の結果を踏まえて，高速インターコネクトの性能を最大限生かすような遠隔メモリスワッピングの設計を検討する．

本システムは，既存の Linux スワップ機構が想定している環境とは以下の点で異なる．

- スワップ先はネットワーク経由によるメモリサーバである．
- 遠隔ホストとメモリサーバの全体で 1 つの巨大メモリ空間を利用するアプリケーションが実行されると想定する．メモリサーバ上のメモリ全てを使わないと動作しないアプリケーション実行を仮定しているので，それ以外のアプリケーションが同時に実行する状況は想定しないからである．
- アプリケーションは科学技術計算分野を想定している．これらアプリケーションはメモリのアクセスパターンが比較的静的に決まる．あるいは，メモリのアクセスパターンは実行ごとに違うことはなく，常に同じようなアクセスパターンである．

このように単純化すると，分散共有メモリシステムが提供する一貫性プロトコルも必要とせず，以下のような単純機能が実現されれば良いことになる．

### 4.1 物理メモリの扱い

連続物理メモリを単位とする．アプリケーションの持つワーキングセットサイズと，使用するネットワークのメッセージサイズと通信性能の関係から，最適な一回のスワップサイズを決める．

### 4.2 スワップの契機

物理メモリは常に枯渇状態なのでスワップ機構がデーモンとして実現されている必要はない．さらに，予備実験の結果にあるように，デーモンによる実装は，デーモンがいつ実行されるかは CPU スケジューラのスケジューリングアルゴリズムと他のシステムプロセスの状態に依存するために振舞が予測できなくなる欠点がある．

### 4.3 スワップ対象ページの選定

考察で述べたように，メモリサーバを使用する場合は，スワップ領域のページ連続性よりもスワップのためのデータ転送数を減らすことが重要である．そこで，スワップ対象ページは，更新されたページのみとする．メモリアクセスパターン指標に基づき，投機的にページのスワップイン/スワップアウトを行う．

### 4.4 実装の概略

上記で述べた機構を Linux の character デバイスで実現する場合の概略を以下に示す．

(1)　mmap システムコールを実装する．アプリケーションが使用する物理ページを確保する．

(2)　ページフォルトハンドラを実装する．ページフォルトの契機でスワップ機構を実行する．

(3) スワップ対象ページのヒント情報を得るための ioctl システムコールを実装する.

アプリケーションプログラムを変更しなくても，プログラムのデータ領域が上記 character デバイスでマッピングされるためには，メモリ領域を確保しているシステムライブラリを変更する必要がある．malloc ライブラリは最終的には，mmap システムコールライブラリを呼ぶ．そこで，mmap システムコールライブラリを置き換えて，対象となるメモリ領域が mmap される時に上記 character device が呼ばれるようにすれば良い．

## 5. 関連研究

Iftode らの研究[3] や GMS[1] など，遠隔ホストのメモリを利用したページングシステムに関する研究は比較的古くから行われている．Ethernet LAN 上のクラスタで遠隔ホストのメモリをスワップとして利用する研究には Remote Mamory Pager (RMP)[6] や Network RamDisk[2] などがある．これらの研究は，高性能というよりは高信頼性の遠隔スワップメモリシステムを構築することを目的としており，通信には TCP/IP が用いられている．また，NSwap[8] は TCP/IP・UDP/IP を用いて遠隔スワップメモリを実現する Linux のカーネルモジュールである．RMP，Network RamDisk，NSwap はいずれも，10Base-T や 100Base-T といった遅いネットワークを用いて実験が行われている．

高速インターコネクトを用いて遠隔スワップメモリを実現しようとする研究としては，Infiniband[4] を用いた HPBD[5]，10Gb Ethernet NIC UZURA[9] を用いた NUZURA[12]，及び Myrinet Express を用いた DMPS[11] がある．これらはいずれも，遠隔ホストと通信して読み書きを行うような Linux 用のブロックデバイスを作成し，スワップデバイスとしてそのデバイスを用いることにより，より高速なスワッピングを実現している．また，金井ら[10] は，DIMMNet-2[13] を用いて Windows 上に分散共有メモリシステムを構築した．

## 6. おわりに

ネットワークブロックデバイスを開発することにより，現状の Linux スワップ機構を変更しなくても遠隔スワップメモリシステムを実現することが可能である．しかし，現状の Linux の問題点を考察した．

考察から character デバイスによるメモリマップ機能およびページフォルト機能を使用した遠隔スワップ用デバイスを設計した．今後，本論文で設計した character デバイスを実装し評価していく．

## 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) Feeley, M.J., Morgan, W.E., Pighin, F.H., Karlin, A.R., Levy, H.M. and Thekkath, C.A.: Implementing Global Memory Management in a Workstation Cluster, *Proceedings of Symposium on Operating Systems Principles*, pp.201–212 (1995).
2) Flouris, M. and Markatos, E. P.: The Network RamDisk: Using remote memory on heterogeneous NOWs, *Cluster Computing: The Journal on Networks, Software and Applications*, Vol.2, No.4, pp. 281–293 (1999).
3) Iftode, L., Li, K. and Petersen, K.: Memory Servers for Multicomputers, *Proceedings of the 38th IEEE International Computer Conference*, pp.534–547 (1993).
4) Infiniband Trade Association: `http://www.infinibandta.org/home`.
5) Liang, S., Noronha, R. and Panda, D.K.: Swapping to Remote Memory over InfiniBand: An Approach using a High Performance Network Block Device, *2005 IEEE International Conference on Cluster Computing* (2005).
6) Markatos, E.P. and Dramitinos, G.: Implementation of a Reliable Remote Memory Pager, *USENIX 1996 Annual Technical Conference*, pp. 177–190 (1996).
7) Myricom, Inc.: Myrinet, `http://www.myri.com/`.
8) Newhall, T., Finney, S., Ganchev, K. and Spiegel, M.: Nswap: A Network Swapping Module for Linux Clusters, *Proceedings of Euro-Par '03 International Conference on Parallel and Distributed Computing*, pp.1160–1169 (2003).
9) 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 10Gb Ethernet インタフェースカードの設計 (ネットワーク)，情報処理学会論文誌，Vol.47, No.12, pp.74–85 (2006).
10) 金井　遵，森　拓郎，荒木健志，田邊　昇，中條拓伯，並木美太郎：主記憶以外に大容量メモリを有するメモリ/ネットワークアーキテクチャ，*SACSIS*, pp.419–426 (2006).
11) 今井照之，松葉浩也，石川　裕：分散ページングによる大規模仮想メモリ空間，情報処理学会研究報告，Vol.2007, No.17, pp.85–90 (2007).
12) 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 (CPSY，コンピュータシステム)，Vol.106, No.287, pp.7–12 (2006).
13) 北村　聡，濱田芳博，宮部保雄，宮代具隆，伊沢　徹，田邊　昇，中條拓伯，天野英晴：DIMMnet-2 ネットワークインタフェースコントローラの設計と実装，情報処理学会論文誌，Vol.46, No.SIG 12, pp.13–26 (2005).