社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report

遠隔メモリスワップのためのユーザレベルソフトウエアDLMの性能評価

緑川 博子† 黒川 原佳‡1 姫野 龍太郎‡2

†成蹊大学 〒180-8633 東京都武蔵野市吉祥寺北町 3-3-1
‡理化学研究所 ‡1 情報基盤センター, ‡2 次世代計算科学研究開発プログラム

〒352-0198 埼玉県和光市広沢 2-1

E-mail: †midori@st.seikei.ac.jp, ‡{motoyosi, himeno}@riken.jp

あらまし 64bitOS の普及により，ユーザが大きなアドレス空間を使えるようになってきた．そこでローカル物理メモリサイズに制限されず，クラスタの各ノードの遠隔メモリを集めて仮想的に大容量メモリとする分散大容量メモリシステム DLM とそのコンパイラをユーザレベルソフトウエアのみで構築した．本報告では，10GbEthernet, TCP 実装で，STREAM や Himeno ベンチマークを用い，プログラムレベルの遠隔メモリアクセス性能，応用プログラムにおける性能を調べた．この結果，既存の kernel の swap 処理とは独立にユーザソフトで DLM を構築し，大きなデータサイズ単位で遠隔ホストとのやり取りを行うことによって，特殊なハードウエア，プロトコル，デバイスドライバ構築，kernel の改変などを用いたものにも匹敵する性能と安定した稼動環境を得ることができた．

キーワード クラスタコンピューティング， 大容量メモリ，遠隔メモリ，分散メモリ，ページスワッピング

The Performance of DLM:User-Level Software for Remote Memory Swapping

Hiroko MIDORIKAWA † Motoyoshi KUROKAWA ‡ and Ryutaro HIMENO ‡

† Department of Computer and information science, Seikei University, 3-3-1, Kichijoujikita-machi, Musashino-shi, Tokyo 180-8633, Japan

‡ Advance Center for Computing and Communication, RIKEN The Institute fo Physical and Chemical Research, 2-1, Hirosawa, Wako-shi, Saitama, 351-0198, Japan

E-mail: † midori@st.seikei.ac.jp, ‡ {motoyosi, himeno}@riken.jp

Abstract Emerging 64bitOS's drastically enlarge available memory address space and open the door to new applications using very large data. In this background, authors designed Distributed Large Memory System: DLM and its compiler, which gives us very large virtual memory using remote memory distributed over cluster nodes. In this paper, the performance of the DLM programs using remote memory is compared to the ordinary programs using local memory. The results of the benchmarks, STREAM and Himeno BMK, reveal that the DLM obtains comparable or better performance than other remote paging schemes using a block swap device to access remote memory. The advantages of DLM are not limited in performance but also in easy availability and high portability, because it is only user-level software with no special hardware. To gain high performance, the DLM can tune its parameters independently from kernel swap parameters. It is free from the overhead of kernel swap daemon and provides stable running.

Keyword Cluster Computing, Large Memory, Remote Memory, Distributed Memory, Page Swapping

1. はじめに

64bit の OS や CPU の普及により，桁違いに大きなアドレス空間（X86_64 現実装でも 256TB）を使えるようになってきた．しかし，１台のコンピュータで提供できる物理メモリにはスロット数などのハードウエア制約もあり，大容量物理メモリ搭載マシンは非常に高価になる．物理メモリ不足時には swap 領域（通常，ローカルディスクのファイル)を大容量化することにより，大データを扱うプログラムの実行は可能だが，実メモリにデータが全て収まる場合に比べ非常に低速になり，多くの場合，実際の使用に耐えない．この原因は単にハードディスクのシーク性能や入出力帯域の不十分さだけに拠るのではなく，OS kernel におけるスワップ処理のソフトウエアオーバヘッドや動作不安定性の現状なども一つの原因になっていると考えられる．

一方，ネットワーク上の遠隔メモリを有効利用しようとする研究は古くからあるが，最近ではネットワークの高速化を背景に遠隔メモリをページングに用いる

試みがされてきている．現状では， kernel の変更は最小限に抑え，遠隔メモリ利用のためのブロックデバイスドライバを構築してスワップデバイスとして使う試みなどが多くなされている[3]-[7]．

一方，我々は，遠隔メモリ利用技術の一つとして，ソフトウエア分散共有メモリなどでも古くから使われているユーザレベルプログラム（ライブラリ）による実装方式を用い，ネットワークにつながれたホストの比較的小容量の遠隔メモリを集めて，仮想的に大容量メモリとして利用することが可能な分散型大容量メモリシステム DLM (Distributed Large Memory) を構築した[1][2]．また，独自のコンパイラと API を提供することにより，従来の逐次 C プログラムを大幅に変更することなく，ユーザには最小限の負担で遠隔メモリを利用できる環境を構築した．

OS の swap デバイスを変更する前述の手法と異なり，DLM における遠隔メモリ利用は，OS swap 機構や swap デバイスを代替するものではなく，OS の swap とは独立である．この手法によるユーザプログラムの遠隔メモリ利用は，ユーザが自由に決めたローカルメモリ利用サイズ内で，通常，メモリに多少の余裕がある正常実行状態においてプログラムが実行される．このため，kernel swap デーモンなどによるソフトウエアオーバヘッドを生じない上，メモリ枯渇状態における様々なトラブル[7]が生じにくい．また kernel スワップ機構とは独立であるため kernel 自体の従来の swap パラメタや特性に支配されず， swap ページサイズも自由に設定が可能で，場合によっては，どのデータをローカル固定でおきたいかを応用ごとにユーザがプログラム上で指示することも可能である．

すでに 1GbE および 10GbE 結合クラスタにおけるテストプログラム評価では，通常 OS の swap ファイル（ローカルハードディスク）利用時に比べ，1GbEther クラスタであっても，遠隔メモリに展開してアクセスしたほうが有利であることがわかっている[1]．

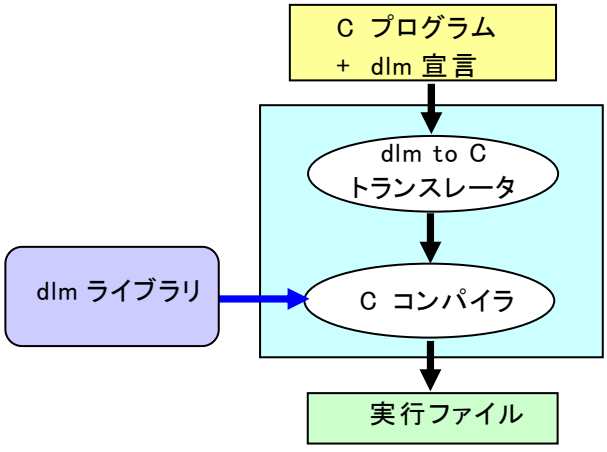

図１　コンパイラの構成図

本報告では，ローカルメモリのみを使う通常プログラムと遠隔メモリを一部用いた DLM プログラムとで，どの程度の性能低下があるか，10GbE 結合のクラスタ上で調べた．すでに，単純な配列の連続アクセスや NPB の FT, IS, CG についての性能評価を行っているが[2]，本報告では，マイクロベンチマークとしてメモリアクセス帯域を調べる STREAM ベンチマーク[8]を用いて，TCP を用いた本実装での遠隔メモリアクセスの性能を調べた．また計算やデータアクセス局所性を含む応用例として Himeno ベンチマーク[9]の性能を調べた．

2. DLM ユーザインターフェースとコンパイラ

DLM によって遠隔メモリを利用するには，以下のような静的な dlm データ宣言，もしくは，dlm_alloc による動的割付けを用い，遠隔メモリに展開したいデータをユーザが指定する．

```
例１　dlm double array1[Size1][Size2][Size3];
例２　int *array2;
      array2= ( int *) dlm_alloc ( sizeof(int) * SIZE );
```

DLM コンパイラ(dlmc)は，ユーザには透過的に dlm ライブラリ関数を挿入し，逐次 C プログラムを， ローカルホストにおける計算プロセスと遠隔ホストにおけるメモリサーバプロセスから成る並列プログラムに自動変換する[1][2]．DLM コンパイラは，図１のように DLM 宣言を通常 C プログラムへ変換するトランスレータと C コンパイラから成り，現実装では後段には gcc を用いて，dlm ライブラリとリンクしている．

この DLM コンパイラを使うと，既存プログラムから DLM を用いるプログラムへの変更が容易にできる．図２は，本報告で用いた Himeno ベンチマークのオリジナルプログラムの変更部分を示している．オリジナルの C プログラムでは，用いる配列が大域変数として宣言されている．元の static は，ソースファイルが１つなので C では本来，不要で，DLM 版では単に dlm を付加している．これ以外のコードの変更は必要ない．

同様に，STREAM ベンチマークの変更点を図３に示す．こちらは，後述する理由で動的メモリ割り当てに変更しているが，図２のような大域変数として静的宣言を用いることも可能である．この例でもほとんど変更はなく，ユーザが従来のプログラムからの大きな変更なしに遠隔メモリを利用できることがわかる．

3. DLM システム

DLM システムは，前述の DLM 用 C プログラムと DLM 設定ファイルを用意すれば誰にでも利用できる．

DLM 設定ファイルは，プログラムを実行するホスト名を先頭行に，メモリを提供できるメモリサーバホ

```
Original Version
      static float    p[MIMAX][MJMAX][MKMAX];
      static float    a[4][MIMAX][MJMAX][MKMAX],
                      b[3][MIMAX][MJMAX][MKMAX],
                      c[3][MIMAX][MJMAX][MKMAX];
      static float    bnd[MIMAX][MJMAX][MKMAX];
      static float    wrk1[MIMAX][MJMAX][MKMAX],
                      wrk2[MIMAX][MJMAX][MKMAX];

DLM Version   （下線 dlm を付加）
      dlm float   p[MIMAX][MJMAX][MKMAX];
      dlm float   a[4][MIMAX][MJMAX][MKMAX],
                  b[3][MIMAX][MJMAX][MKMAX],
                  c[3][MIMAX][MJMAX][MKMAX];
      dlm float   bnd[MIMAX][MJMAX][MKMAX];
      dlm float   wrk1[MIMAX][MJMAX][MKMAX],
                  wrk2[MIMAX][MJMAX][MKMAX];
```

図２　DLM 用 Himeno ベンチマークへの変更点

```
Original Version
   static double    a[N+OFFSET],
                    b[N+OFFSET],
                    c[N+OFFSET];

DLM Version   （ポインタと動的わりあてに変更）
double *a,*b,*c;   //大域変数として宣言

 //以下は main プログラムの先頭部分に挿入
 a=(double *)dlm_alloc((N+OFFSET)*sizeof(double));
 b=(double *)dlm_alloc((N+OFFSET)*sizeof(double));
 c=(double *)dlm_alloc((N+OFFSET)*sizeof(double));
```

図３　DLM 用 STREAM ベンチマークへの変更点

スト名を次行から書いたファイルで，各ホストで利用できるメモリ量(MB)も記述したファイルである．図４に DLM 設定ファイル例(hostfile)とそれに対応した DLM システムのプロセス構成図，及び，プログラム実行コマンドの例を示す．コマンドでの--以降で DLM へのパラメタを指定し，-f で DLM 設定ファイルの hostfile を指定，-n で計算ホストとメモリホストの総和ホスト数，４ホストでの実行を指定している．

DLM システムは，起動時に指定ホストにメモリサーバプロセスを立ち上げ,指定メモリサイズ分を DLM 計算プロセスの要求に応じて提供するように設定する．現実装では，設定ファイルの先頭行から順にメモリを使用していき，提供メモリが足りなくなると，次行のメモリサーバを用いるようにしている．メモリサーバプロセスの遠隔生成，計算ホストにおける通信スレッド生成，遠隔メモリアクセス，ユーザプログラム終了時の遠隔プロセスやソケットの終了処理などは，すべてユーザには透過的に行われる．

遠隔メモリとローカルメモリのデータ交換単位は，DLM ページと呼ぶ単位で行っており，DLM システムにおけるメモリ管理の単位としても用いている．

DLM ページサイズは DLM システム構築時に変更可能で，OS のメモリ管理単位であるページサイズの整数倍で自由に設定できる．

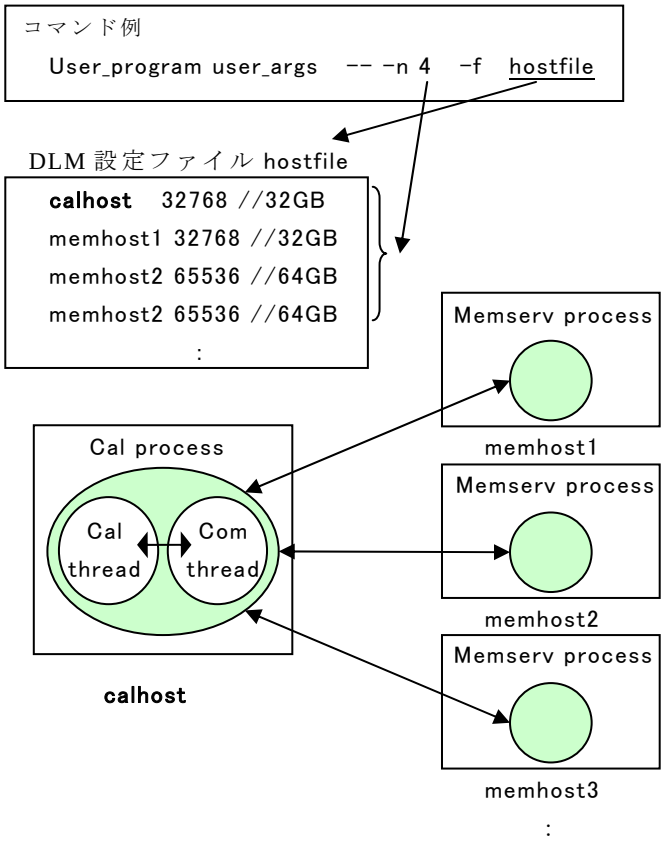

図４　DLM 設定ファイルとプログラム実行例

4. 10GbEther クラスタにおける性能評価

今回の実験で用いたシステム環境を表１に示す．

4.1. マイクロベンチマークによる性能

定常的なメモリアクセス帯域を調べるためのベンチマークである STREAM[8]を用い，DLM システムによる遠隔メモリアクセスの帯域を測定した．STREAM，及び STREAM2 は，表２に示すように典型的な配列アクセス操作を配列全体に対して複数回繰り返し行い，様々な影響を受ける１回目の測定値を除き，繰り返し中の最良値を性能結果とする．通常，主メモリの帯域を調べる場合には，配列サイズを十分に大きくとり，キャッシュアクセスの影響を抑える必要がある．今回はローカルメモリを越える遠隔メモリにアクセスする場合の計測をすることから，１つの配列サイズは，後述のローカルメモリ性能の計測から 100M 個(double)とし，全体のメモリ使用量（STREAM では配列 3 個で 2.4GB，STREAM2 では配列２個 1.8GB）のうち，

ローカルメモリの占める割合を 8%程度とし 92%を遠隔メモリに割り当てて測定した．また DLM ページサイズを 4KB から 1024KB まで変化させて計測した．

表 1　10GbEther クラスタ　(理研 CSLM クラスタ)

Cluster	HP DL585 G2 x 5 Nodes
Node CPU	DualCore AMD Opteron(8220SE) 2.8GHz x 4 (8Cores)
NodeMemory	64GiByte(67.1GB)
PCI bus	64bit/100MHz PCI-X, PCI-Expressx4 PCI-Expressx8
OS	Linux　kernel 2.6.9-42　x86_64
Compiler	gcc version 3.4.6
Network	10GbE protocol (Myri-10G)
Switch	Fujitsu XG1200(10GbE Switch)
Hard Disk	SAS　147GB 10krpm　2 台　RAID1 Smart array 5i

STREAM は公開されている C プログラム版のままで配列を静的宣言するものと（stream_static 版）と，静的宣言した配列を動的メモリ割り当てに変更した版（stream_malloc 版）を作成して用いた．元々の static 版では，現在 gcc でコンパイルすると bss サイズが制限されており，配列サイズ 130M 個程度より大きいサイズの静的配列は使用できず，大きなデータを用いるには動的割り当てを使うことになるためである．また，static 版は，コンパイラ最適化か，実行時の影響によるのか，配列アクセス性能が malloc 版に比べ若干高速であるため，動的割当てを用いている DLM との比較には，公平性のため stream_malloc 版の性能を基本とした．STREAM2 の C プログラム版は公開されていないので，上記で作成した STREAM C 版の kernel 部分を表 2 のものに換えて作成した．また，ベンチマーク内で性能計測後に行う Verification で用いる総和変数の double や，配列サイズや for 文インデックスなどの int は，size_t や long double に変更し誤動作が起きないように 64bitOS に適合させている．

まず DLM を使用しない通常プログラム（すべてローカルメモリを使用）の場合の STREAM（malloc 版）の性能を図 5 に示す．横軸は 3 つの配列のメモリ総量（Byte）を示し，STREAM プログラム中の配列サイズ N が 10k～2G 個の範囲に対応する．

この中で，キャッシュの影響を受けず性能が安定している総計 2.4GB のメモリ量（配列サイズ 100M 個に相当）の時の STREAM malloc 版と static 版の性能を表 3 に示す．一方，2.4GB のうち約 8%（200MB）のみをローカルメモリにおいた DLM での実行結果を図 6 に示す．DLM ページサイズが 1024KB と 4 KB の場合，バンド幅はそれぞれ約 370MB/sec と 40MB/sec である．表 3 の通常プログラムのローカルメモリの場合と比較すると，図 7 に示すように，ローカルメモリアクセスに比べ，7 倍から 70 倍，低速になる．

表 2　STREAM, STREAM2 ベンチマーク

	Kernel	Code
STREAM	COPY	a(i) = b(i)
	SCALE	a(i) = q*b(i)
	ADD	a(i) = b(i) + c(i)
	TRIAD	a(i) = b(i) + q*c(i)
STREAM2	FILL	a(i) = q
	COPY	a(i) = b(i)
	DAXPY	a(i) = a(i) + q*b(i)
	SUM	sum = sum + a(i)

この結果は，配列の 2 回目以降のアクセスではアクセスデータが遠隔メモリにスワップアウトされており，遠隔メモリアクセス性能にほぼ対応する．次にローカルメモリと遠隔メモリの比率を変化させ，性能を調べた．COPY と TRIAD の結果を図 8 ,図 9 に示す．横軸はローカルメモリ比率で縦軸は通常プログラム(100%ローカルメモリ)と比較した性能低下度である．COPY はローカルメモリが 70%を切ると急激に性能が悪くなるが，これは STREAM の 3 つの配列のうち 2 つしか使用しないため，これらがローカルメモリに収まっている間は，通常プログラムと同じ性能になっていることを示す．70%を超えると，DLM ページサイズに応じて性能は悪化する．この処理ではページサイズが 4KB から 16KB の時，性能が悪いが，64KB を超えると 1024KB との差は少ない．TRAID では，ローカルメモリ率の高い領域からの性能低下が見られるが，これは 3 つの配列を用いるためである．

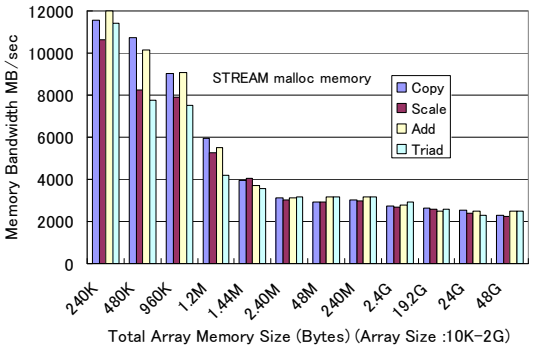

図 5　通常　ローカルメモリバンド幅（STREAM malloc）
Array Size　10K～2G　ローカルメモリ率 100%

表 3　通常　ローカルメモリバンド幅（MBytes/sec）
ArraySize100M（2.4GB）　ローカルメモリ率 100%

STREAM　Array Size:100M 2.4GByte

	Copy	Scale	Add	Triad
static	2976	2804	2926	3153
malloc	2718	2694	2767	2925

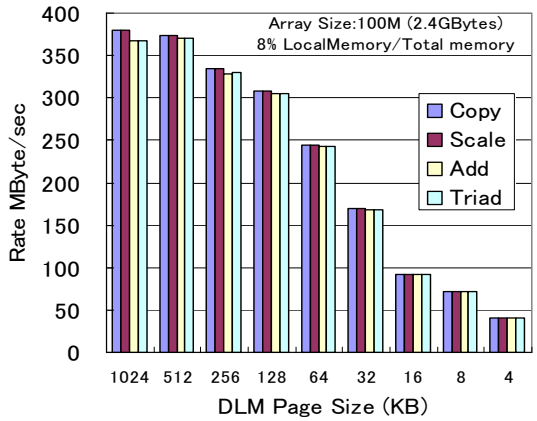

図 6　DLM 使用時　遠隔メモリバンド幅（STREAM）
Array Size　100M　(2.4GB)　ローカルメモリ率 8%

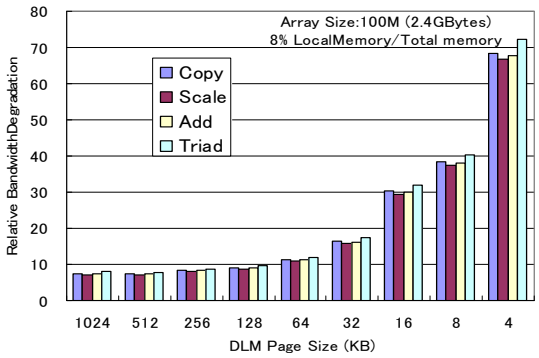

図 7　通常時（ローカルメモリ 100%）に対する性能低下度
Array Size　100M　(2.4GB)　ローカルメモリ率 8%　DLM

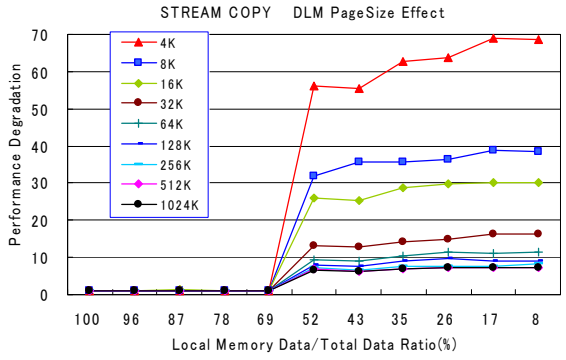

図 8　通常時（ローカルメモリ 100%）に対する性能低下度 STREAM(COPY)

STREAM2 の SUM と DAXPY の性能低下の度合いを図 10 と図 11 に示す．それぞれ STREAM の COPY と TRIAD と同様な傾向を示すが，表 2 に見るように kernel 演算でアクセスする配列が少ないので，性能低下は SUM で最悪でも 50 倍，DAXPY では 70 倍程度となり，COPY の 68 倍と TRIAD の 72 倍よりも性能低下は少ない．FILL も SUM と同様の傾向を示す．

このように，プログラムで確保したデータ領域全体のうち一部のみを使って計算するようなアクセスパターンであれば，性能低下を低く抑えることが，ある程度可能である．またどの場合も DLM ページサイズを 16KB 以下にすると性能低下が著しい．

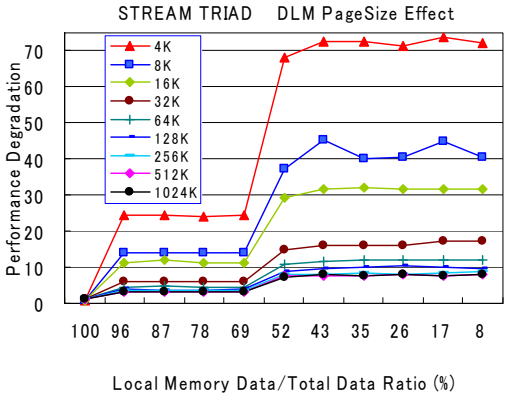

図 9　通常時（ローカルメモリ 100%）に対する性能低下度 STREAM(TRIAD)

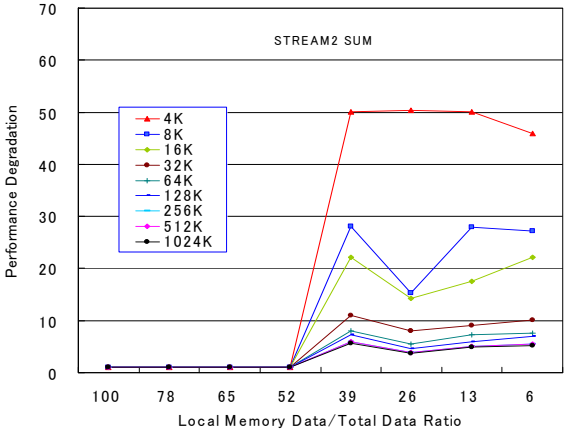

図 10　通常時（ローカルメモリ 100%）に対する性能低下度 STREAM2(SUM)

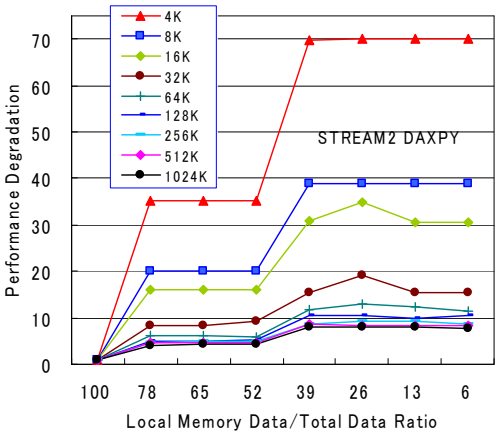

図 11　通常時（ローカルメモリ 100%）に対する性能低下度 STREAM2(DAXPY)

4.2. 応用プログラムにおける性能

実際の応用プログラムでのメモリアクセスパターンと計算/データアクセスの比率において，どの程度の

性能が得られるか，Himeno ベンチマーク[9]を用いて調べた．このベンチマークは，非圧縮流体解析処理の性能評価のために，ポアッソン方程式解法をヤコビの反復法で解く場合の主要ループの処理速度を計るものである．メモリアクセス負荷の高いベンチマークで多重ループ処理で配列全体をスキャンする．ここでは C プログラム版の Large サイズ（257x257x513 サイズ，1.9GB）を用いた．このベンチマークの性能は MFLOPS で出力されるので，この値を通常プログラム（ローカルメモリ 100%）の場合と比較し，DLM ではどの程度性能低下したかを調べた．

図 12 はローカルメモリの比率と DLM ページサイズを変えて実行した性能低下の結果である．性能低下は DLM ページサイズにも依存するが，ローカルメモリ率が 10%で DLM ページサイズ 4 KB のときの最悪値でも 35 倍程度の低下で，前節のマイクロベンチマークの 70 倍の結果に比べると，負荷は軽いといえる．実際の応用，特に HPC 関連処理では，メモリアクセス以外の一定量の計算処理が存在し，また程度の差はあってもアクセス局所性があるため，前節のような人工的かつ過酷なマイクロベンチマークの状況は少ないと考えられる．

このプログラムでは，ローカルメモリの比率が 50%程度で，4KB の DLM ページサイズの場合 30 倍に性能が低下しており，既に測定した NPB の FT,CG,IS の結果[2]と比べても負荷が高い応用といえる．この結果は，[4]などの専用高速 NIC とブロックデバイスドライバを用いた場合の Himeno BMT の性能結果とも一致している．しかし，DLM ではページサイズが 1024KB の時，ローカルメモリ比率が 10%であっても速度低下は 5 倍以下であった．また，DLM ページサイズが 64KB 以上ならば，ローカルメモリが 10%であっても 7 倍程度の速度低下で済むことがわかる．

Himeno ベンチマークは，[4][5]などで評価に用いられている qsort や，一部の NPB(lu，sp など)に比べ，ループ内で広範囲なデータアクセスが発生して多数の swap を引き起こす，遠隔メモリアクセス率が高い応用と言われている[4]．しかし，OS kernel の swap 単位である 4 KB に限定されずに，大容量メモリ空間にふさわしい大きなページ単位で swap することにより，速度低下を抑えて遠隔メモリを利用することができることがわかる．

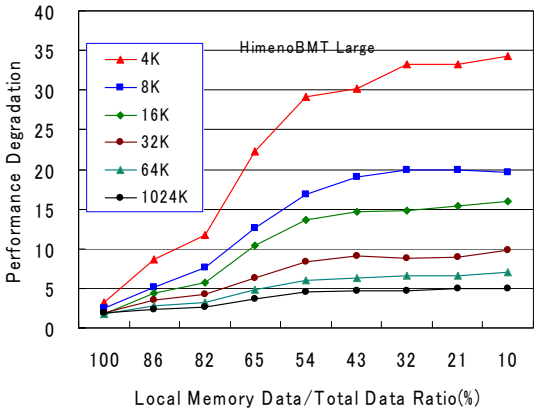

図 12　通常時（ローカルメモリ 100%）に対する性能低下度　HimenoBMT(Large)

5. おわりに

ユーザの負担を最小限にしながら，遠隔メモリを利用できる環境を，ユーザレベルソフトウエアを用いて構築した．DLM は，汎用のプロトコル，汎用 NIC を用いて，特殊なハードウエア，プロトコル，デバイスドライバ構築，kernel の改変をせずとも，従来のブロックスワップデバイス置き換えと高速専用 NIC などを用いた手法にも匹敵する性能とより安定した稼動環境を得ることができた．

今回のシングルクライアント版とは別に，現在，マルチクライアント向けメモリサーバ版（DLM-M）も構築，評価中である．今後，さらに大規模なメモリを使う応用に関して評価を行う予定で，これと並行して実装の細部の評価検討，改善を加える予定である．

謝辞

日頃より研究活動を支援頂いている成蹊大学理工学部情報科学科甲斐宗徳教授に深謝致します．

文　献

[1] 緑川,小山,黒川,姫野, "分散大容量メモリシステム DLM の設計と DLM コンパイラの構築", 信学会 CPSY,信学技報 Vol.102, No.398, pp.29-34,2007

[2] 緑川，黒川,姫野, "遠隔メモリを利用する大容量メモリシステム DLM とコンパイラ", 情処学会，研究会資料 HPC115-7, pp.37-42, 2008

[3] 北村，松葉，石川，“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，” 情処学会，研究会資料,HPC-111(21), pp.121-126, 2007.

[4] 後藤，佐藤，中島，久門，“10GbEthernet 上での RDMA を用いた遠隔スワップメモリの実装，” CPSY，信学技報告 Vol.106 No.287, 200

[5] S. Liang, R. Noronha, and D. K. Panda,, Swapping to Remote Memory over InfiniBand: An Approach using a High Performance Network Block Device, IEEE Cluster Computing, 2005

[6] Tia Newhall et al. “Nswap: A Network swapping Module for Linux Clusters”, EuroPar03, 2003

[7] Pavel Mache, Linux Network Block Device, (1997) http://nbd.sourceforge.net/

[8] STREAM Benchmark http://www.cs.virginia.edu/stream/ref.html

[9] HimenoBMT（姫野ベンチマーク） http://accc.riken.jp/HPC/HimenoBMT/index.html