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

信学技報
TECHNICAL REPORT OF IEICE

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

緑川 博子† 小山 浩生† 黒川 原佳‡ 姫野 龍太郎‡*

†成蹊大学 工学研究科 〒180-8633 東京都武蔵野市吉祥寺北町 3-3-1
‡理化学研究所, ‡情報基盤センター, ‡*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, h_koyama@athene.is.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構築し，初期評価を行ったので報告する．この結果，行列ベクトル積プログラムなどで，1Gb／10Gb Ether 結合クラスタで，swap ファイルを使う通常プログラムの 5 倍～10 ほどの性能が得られた．

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，遠隔スワップメモリ

The Design of Distributed Large Memory System DLM and DLM Compiler

Hiroko MIDORIKAWA† Hiroki KOYAMA† 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, h_koyama@athene.is.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 enable us to use very large virtual memory using remote physical memory distributed in network-connected computers. Initial performance evaluation shows that DLM programs on cluster with 1GbEthernet perform 5 times faster than ordinary programs using local swap file. On a cluster with 10GbEthernet, the DLM program performance reached over 10 times faster than ordinary programs. The DLM's advantages are not limited in performance but also in easy availability, because it only uses user-level software and it needs no modification of OS system parameters and no special hardware.

Keyword Cluster Computing, Large Memory, Remote Memory, Distributed Memory, Remote Swap Memory

1. はじめに

64bit の OS や CPU の普及により，32bitOS の 4GB アドレス空間から，桁違いに大きなアドレス空間（現状実装でも 256TB テラバイト）を使えるようになってきた．しかし，大容量物理メモリを持つシステムが利用できない場合や，swap 領域サイズが足りない場合には，大容量メモリを使用するプログラムを事実上，実行することができない．swap ファイルを大容量化することにより大きなプログラムを実行させることが可能であるが，一般ユーザとっては，システム構築時に設定されたパラメタを個々の希望に応じて安易に変更してもらうことが難しいことも多い．

このような環境でも，安価なクラスタの各計算ノードの比較的小容量の物理メモリを集めて遠隔メモリとして利用し，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 を構築した．さらに，従来の C プログラムからの変更をほとんど行わずに，通常の C プログラムを簡単に DLM システムで実行できるように，DLM コンパイラを構築した．本報告では，その DLM システムの構造と，DLM コンパイラと API，初期試作システムの稼働結果，初期性能などを報告する．

2. 分散大容量メモリシステム DLM のユーザインターフェースとコンパイラ

DLM システムは一般ユーザが手軽に使えることを意図し，OS kernel やシステムソフトウエアに手を入れて元の swap システムなどを変更することをせずに，純粋にユーザレベルソフトウエアだけを用いる．ユー

ザの C プログラムに後述する DLM 宣言を加えるだけで，容易に，クラスタに分散したメモリを大容量メモリとして用いることができる．

2.1. 分散メモリデータの静的宣言と動的割り当て

DLM を利用するには，C プログラムの中で，どのデータを DLM に展開するかをユーザが指定する．静的宣言であれば，以下の例 1,2 のように，宣言の前に dlm という予約語を付加するだけである．動的割り付け malloc であれば，関数名を dlm_alloc に変更するだけである．これにより，ユーザは，どの部分のデータをメモリサーバに展開するかを指定でき，それ以外のデータについては，計算ノードのローカルメモリを使うことが保障される．もちろん dlm 指定されたデータであっても，ローカルノードのメモリに余裕があれば，ローカルメモリから順に割り当てて行くので，メモリサーバのメモリにデータ展開や swap するのは，ローカルメモリが足りなくなったときだけである．図 1 は，この宣言を利用して書いた簡単のプログラム例で（行列とベクトルの積）である．C プログラムからの変更部分は，太字の dlm 宣言のみである．

例 1　**dlm** double data[Size1][Size2][Size3];
例 2　int *p:
　　p = (int *) **dlm_alloc**(sizeof(int) * SIZE);

```
#include <stdio.h>
#define N 16384   // total memory 2048MB + 32KB

dlm double a[N][N], x[N], y[N];   // DLM 使用

int main(int argc, char *argv[])
{ int i,j;
  double temp;
  //  行列 a を初期化
  for(i = 0; i < N; i++)
    for(j = 0; j <N; j++) a[i][j] = i;
  //    ベクトル x を初期化
  for(i = 0; i < N; i++)      x[i] = i;

  //  a[N][N]*x[N]=y[N] 計算
  for(i = 0; i < N; i++){
      temp = 0;
      for(j = 0; j <N; j++)   temp += a[i][j]*x[j];
      y[i] = temp;
  }
  return 0;
}
```

図 1　DLM 使用プログラム例（matv.c）dlmc-0.0.3

2.2. メモリサーバ指定ファイル

ユーザは 2.1 で述べた DLM プログラムを作成する他に，メモリサーバとして使うホスト名一覧と，それぞれのホストでどの程度の容量のメモリを DLM システムに使用させるかを記述した設定ファイルを作成し，プログラム実行時に指定する．

設定ファイルの先頭行は計算ノードホストと，計算ホストでローカルメモリとしてどの程度の DLM にメモリを使用させるかということを示す．2 行目以降は，メモリサーバとして提供できるホスト名と提供メモリ容量を記述する．(メモリ容量は現在，MB 単位で記述)

このファイルを実行時に指定すると，DLM システム起動時に，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．現在のところ，設定ファイルに複数のメモリサーバが指定されている場合は，設定ファイルの上位行のメモリサーバホストから順に使用していき，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．すなわち，メモリ使用場所の優先順位を意味する．図 2 のように，メモリサーバと計算サーバのホスト台数を実行時に指定でき，設定ファイルの上位から使用していく．ノード数 1 の指定の場合には計算ノードのみしか使わないという意味となる．

設定ファイルに記述された全メモリサーバの総提供メモリ容量以上にメモリが要求された場合には，ユーザプログラムにエラーを返し，DLM システムは終了する．

DLM 設定ファイル例　hostfile

```
calhost      2048   // 2GB
memhost1     8192   // 8GB
memhost2     4096   // 4GB
memhost3     4096   // 4GB
    :
```

プログラム実行コマンド例

prog　-- -n 4 -f hostfile

設定ファイル hostfile の先頭 4 行を用い，計算ノードとメモリサーバノード 3 台を使用する指定例

図 2　設定ファイルとプログラム実行例

2.3. DLM コンパイラ　dlmc

DLM システムは起動時に前述の DLM 設定ファイルを読み込み，実行時にメモリ動的割り当てやメモリサーバへの swap 要求を処理する．このため，DLM コンパイラは，まずメモリサーバプロセスの遠隔起動や計算プロセスの初期設定などを行う初期化関数 dlm_init()をプログラムの開始部分に挿入する．さらに，ユーザにより dlm 指定された静的データ宣言を，動的

割り当てとポインタ表現に置き換える処理を行う．すなわち，プログラムスコープを考慮した変数リネーミングを行う．図４はコンパイル実行例と変換例を示す．

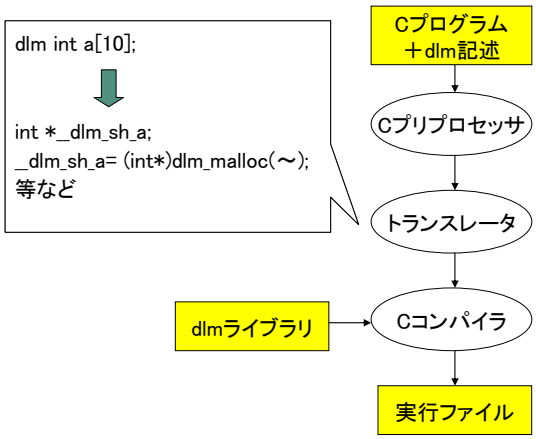

図３　コンパイラの構成図

```
//   sample.c
#define MAX 1000

dlm int a[MAX];

int main(int argc, char *argv[])
{
   int i;

   for(i = 0; i < MAX; i++)    a[i] = i;
   for(i = 0; i < MAX; i++)    printf("%d ", a[i]);
   return 0;
}
```

コンパイル　**dlmc sample.c -ldlm**

```
int (*__dlm_sh_a);

int main(int argc, char *argv[])
{
   int i;

   dlm_init(argc, argv);

   if(MYPID == 0){
      __dlm_dim[0] = 1000;
      __dlm_dim[1] = -1;
      __dlm_div[0] = -1;
      __dlm_sh_a = ( int (*))dlm_mapalloc(__dlm_dim,
__dlm_div, sizeof( int ),0, dlm_nproc);

      for(i = 0; i < 1000; i++)    __dlm_sh_a[i] = i;
      for(i = 0; i < 1000; i++)
         printf("%d ", __dlm_sh_a[i]);
   { dlm_exit(); return 0; }

   }
   dlm_exit();
}
```

図４　DLM コンパイラによるプログラムの変換例

3. DLM システムの構成

3.1.　DLM システムの起動と DLM ページ表

計算ノードホストにおいて，DLM 設定ファイルを指定してプログラムを実行すると，設定ファイルに指定されたホストでメモリサーバプロセスが起動され，計算ノードではユーザ起動のプロセス内で，通信スレッドが生成される．次に計算プロセスとメモリサーバプロセスはソケット(UDP または TCP)を確立させ，使用可能最大メモリサイズ分の DLM ページ表を初期化する．DLM ページ表は，データが割り付けられたページの所在ホスト，ページ内使用サイズなどの情報を保持する．DLM ページのサイズは DLM システム構築時に変更可能であるが，OS のメモリ管理単位であるページサイズの整数倍としている．現在は 4 KB〜32KB 程度での実験を行っている．尚，DLM ページ表は，DLM 設定ファイルに指定された計算ノードの最大 DLM 提供メモリ容量の中から使用するようにしているが，計算ノード内のローカルメモリからメモリサーバへ swap out されないようにしている．

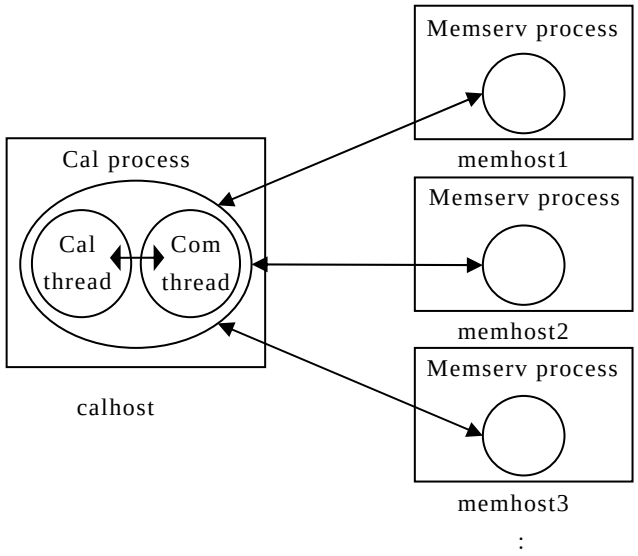

図５　DLM システム構成図

3.2. 通信とエラー処理

DLM は図 5 のように，1 ノードの計算プロセスと 0 ノード以上のメモリサーバプロセス（ノード）から成る．計算プロセスはユーザプログラムの計算を行う計算スレッドとメモリサーバとのやりとりを行う通信スレッドから構成される．メモリサーバプロセスでは計算を行わないので 1 スレッドのみで計算プロセスからのメモリページ割り当て要求，ページ要求，ページスワップ要求などを待つループサーバとしている．

メモリサーバから計算プロセスへのページ転送やエラー通知は，計算プロセス内の通信スレッドが IO 割り込みで受け付ける．メモリサーバとのページの受

信送信を終了すると通信スレッドから計算スレッドへシグナルで知らせる．

ソケットは現在のところ，計算プロセスと各メモリサーバプロセスの通信路のみで，メモリサーバ同志の通信はない．エラー発生時にはエラーを検知したプロセスが計算プロセスに知らせ，計算プロセスがすべてのメモリサーバプロセスにエラーを通知し，終了処理を行う．

3.3. ページ割り当てとページ要求

ユーザプログラムからのメモリ割り当て要求が起こると，計算ノード内のメモリへの割り付けを最優先とし，設定ファイルの記述順の優先度でメモリサーバへの割り付けを行う．

計算スレッドが計算中に，ローカルメモリ以外のデータにアクセスした場合は sigsegv で検知し，そのページを保持するメモリサーバに DLM ページ要求を起こす．その際にローカルメモリに余裕がない場合は，ローカルにある DLM ページから swap ページを選び，該当メモリサーバとの間で要求ページと swap する．現システムでは，swap ページの選択は極力単純化し，ページ表の順に候補を選択している．

4. 性能評価実験

２つの簡単なテストプログラムを用いて，計算ノードの搭載物理メモリよりも大きいサイズのデータ領域を使用するプログラムの実行時間を調べ，ローカルディスクの swap ファイルを使用する通常の C プログラム（SF)と，DLM メモリサーバを使用するプログラム（DLM)とで性能を比較した．使用したプログラムは図 1 に示す行列 a とベクトル v の積を計算する matv と，整数一次元配列への書き込みを行う test0 である．test0 は，先頭から各要素へ連続書き込み後，再度先頭か 1024 要素間隔の離散書き込みを行う．これは，LinuxOS のメモリ管理単位である１ページ（4KB)毎の１整数書き込みに対応する．ここでは UDP 通信， DLM ページサイズは 32KB としている．

4.1　1 GbEther クラスタにおける性能評価

安価で広く用いられている 1GbitEthernet のクラスタにおける DLM システムの効果を調べた．用いたクラスタは表 1 に示すもので，計算ノードの物理メモリは 1GB，swap 領域は 4GB である．DLM 設定ファイルには各メモリサーバが物理メモリサイズより小さめの 750MB 程度を使用する設定にしてある．

表 2 に matv のサイズ N を 12K から 25K（行列 a サイズ 1.2～5.0GB)まで変化させた時の各部分の実行時間を示す．**SF** matv は，従来の swap ファイル(SF)を使用する C プログラム(図３の dlm を削除したもの）である．DLM 版と比較すると，配列 a と x の初期化時の連続アクセス部分の実行時間の差は少ないが，乗算部分の不連続アクセス部分の差が大きい．図 6 は，SF に対する DLM プログラムの速度向上比である．図 7 は実行時間に占める配列 a,配列 x の連続書き込み時間（灰部分）と a*x の乗算時間（白部分）の占める割合を示す．乗算演算回数は SF も DLM も同じなので，SF ではサイズが大きくなると swap ファイルの使用割合が増え，不連続データアクセス時間が増大していくのがわかる．これはディスクファイルアクセスが不連続アクセスを不得意とする性質に起因している．

表１　1 GbEther クラスタ　(　hp-cluster)

Cluster	HP ML150G2　x　8 Nodes
Node CPU	Xeon 2.8GHz x 2CPU　HyperThread
NodeMemory	1GByte (L2cache 1MB/CPU)
PCI bus	64bit/66MHz　PCI-X, PCI-Expressx4
OS	Linux　kernel2.6.20-1.2320.fc5　x86_64
Compiler	gcc version 4.1.1 20070105
Network	1GbE
NIC	Broadcom 5721 PCI-Express Gigabit NIC
Switch	CentreCom GS924GT(1GbE Switch)
Hard Disk	ST3808110AS 80GB　S-ATA2 3Gbps 7200rpm / 8MB cache

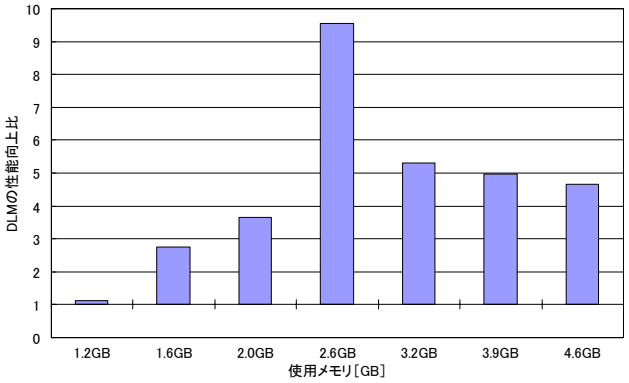

図 6　matv の速度向上比(1GbE クラスタ）

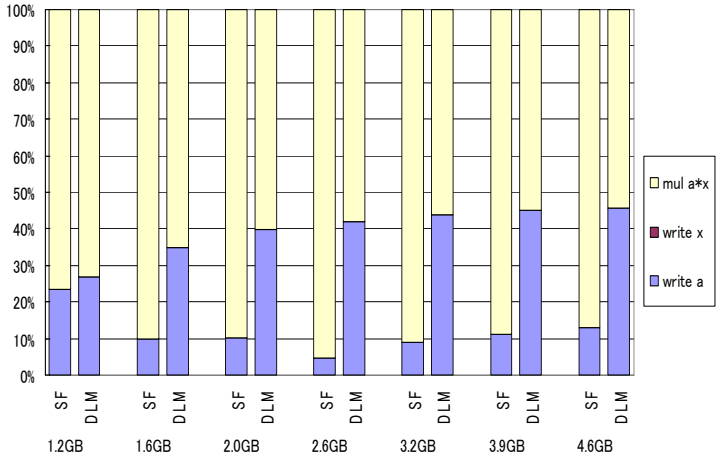

図 7　matv　実行時間に占める初期化と乗算の割合

表２　matv の SF と DLM の比較(1GbE クラスタ)

5.000GB	SF matv	DLM matv -n 7
write a	実行不可能	74.3235
write x		0.004063
mul a*x		86.377434
total		160.704955
4.608GB	SF matv	DLM matv -n 7
write a	88.9981	67.3721
write x	0.0007	0.0034
mul a*x	596.5047	79.6442
total	685.5034	147.0197
3.872GB	SF matv	DLM matv -n 6
write a	67.1970	54.6153
write x	0.0018	0.0035
mul a*x	535.5436	66.6235
total	602.7424	121.2423
3.200GB	SF matv	DLM matv -n 5
write a	46.5746	42.8566
write x	0.0011	0.0029
mul a*x	472.7194	55.0519
total	519.2951	97.9114
2.592GB	SF matv	DLM matv -n 4
write a	33.2976	32.2539
write x	0.0012	0.0029
mul a*x	700.5049	44.4691
total	733.8037	76.7259
2.147GB	SF matvs	DLM matv -n 3
write a	22.6264	24.4745
write x	0.0020	0.0024
mul a*x	201.4762	37.0128
total	224.1046	61.4897
1.568GB	SF matv	DLM matv -n 3
write a	11.2100	14.4606
write x	0.0003	0.0023
mul a*x	102.5674	26.9333
total	113.7777	41.3962
1.152GB	SF matv	DLM matv -n 2
write a	7.1195	7.1809
write x	0.0015	0.0017
mul a*x	23.1196	19.5596
total	30.2405	26.7422

同じく test0 の速度向上比を図８に示す.物理メモリサイズを超えると 5 倍～9 倍程度の速度向上が見られる．DLM プログラムは実行時間のばらつきが 1%未満でほとんどないが, SF の場合には同じプログラムでも実行時間に 2～60%程度の変動がある．このため本報告では SF は計測中の最速値を用いている．DLM プログラムは使用データサイズが増えるにしたがって緩やかに実行時間が増加する．一方 SF は，いずれのテストプログラムでも 2.5GB 程度を使用するあたりで，極端に遅くなるところがある（速度比 9 倍のところ）．SF は, 単純に大容量データを使うほど遅くなるという状況ではない.OS の swap 処理に起因するものなのか, 原因は定かではない.

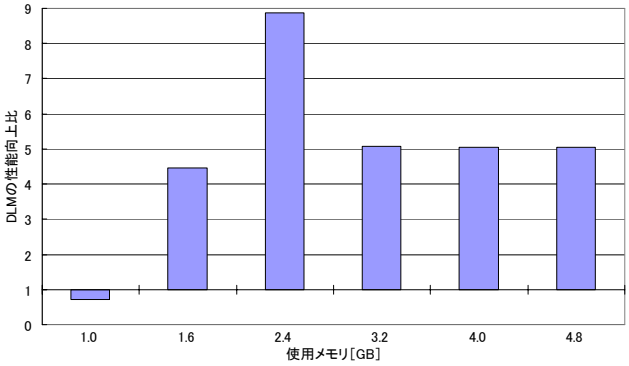

図８　test0 の速度向上比(1GbE クラスタ)

4.2　10GbEther クラスタにおける性能評価

高速通信を持つクラスタとして表 3 の理化学研究所次世代計算科学研究開発プログラムの所有するクラスタ（CSLM)を使用した．ノードの物理メモリは 64GB で，swap 領域を 10GB 持つ．DLM 設定ファイルのメモリサーバが提供するメモリサイズは 60GB とした.

表 3　10GbEther クラスタ　(CSLM)

Cluster	HP DL585 G2 x 5 Nodes
Node CPU	Opteron 2.8GHz x 4 (8Cores)
NodeMemory	64GByte
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
NIC	Myri-10G
Switch	Fujitsu XG1200(10GbE Switch)
Hard Disk	SAS　147GB 10krpm　2 台　RAID1 Smart array 5i

表 4 に 62～76GB 使用時の matv の実行時間を示す. 図 9 に SF に対する DLM の速度向上比を示す．64GB(正確には 67.1GB)の物理メモリを超えない範囲では，SF が swap ファイルを使用せずにすむのに対し，DLM はメモリサーバ間の通信を行っているため，速度向上が 0.25 倍となっている．ただし計算ノードの DLM 提供メモリ容量を 60GB ではなく 64GB にすれば SF と同等レベルとなる. 同様に，67.7GB 使用時には DLM と SF の性能はほぼ等しく 1 で，棒グラフに表れない．物理メモリ 64GB に対し 70.7GB 使用時に 3 倍，73.7GB 使用時に 10.3 倍の性能を得ていて，**4.1** の 1GbE クラスタ（物理メモリの 3 倍の swap ファイル使用）に比べ，

swap利用割合が小さい(swapは物理メモリの15%程度)にもかかわらず，DLM効果が高いことがわかる．この原因は，通信性能が10倍高速であるためと思われる．

表４　matvのSFとDLMの比較(10GbEクラスタ)

76.832GB	SF matv	DLM matv -n 2
write a	実行不可能	154.3412
write x		0.0046
mul a*x		479.8104
total		634.1562
73.728GB	SF matv	DLM matv -n 2
write a	303.8169	135.9808
write x	0.1512	0.0046
mul a*x	5801.8544	453.2156
total	6105.8244	589.2010
70.688GB	SF matv	DLM matv -n 2
write a	195.0827	118.5311
write x	0.2409	0.0045
mul a*x	1489.8479	437.5880
total	1694.1714	556.1236
67.712GB	SF matv	DLM matv -n 2
write a	128.9505	101.4623
write x	0.0009	0.0044
mul a*x	397.6954	420.5199
total	526.6467	521.9866
64.8GB	SF matv	DLM matv -n 2
write a	71.4285	84.8291
write x	0.0011	0.0044
mul a*x	56.2735	415.2058
total	127.7031	500.0393
61.952GB	SF matv	DLM matv -n 2
write a	67.4596	71.9147
write x	0.0010	0.0010
mul a*x	48.1678	54.2941
total	115.6283	126.2098

5. おわりに

DLMシステムとDLMコンパイラを設計構築し，試作システムの簡単な性能評価実験を行った．これにより，swapファイルに展開されたデータに対し，不連続アクセスを含む処理では，1GbEther程度のネットワークでつながれた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た．

２つのテストプログラムでは，1GbEtherクラスタで，遠隔メモリ／搭載物理メモリのサイズ比が2程度で，swap ファイルを使う通常プログラムの５倍ほどの性能が得られた．10GbEtherクラスタでは，遠隔メモリ／搭載物理メモリのサイズ比が0.15倍程度で，10倍ほどの性能が得られた．

また swap ファイル容量に制限がある場合でも，クラスタ物理メモリを集めた容量がそれよりも大きければ，たとえ実行時間が遅くなっても，今まで実行できなかったプログラムが実行できることの意義は大きい．

遠隔メモリをページング/swapに利用するアイデイアは古くからある[1]が，最近では独自に設計された高速通信NICを用いるもの[3]や， OSのswapデバイスとして組み込むもの[2]などが提案されている．本研究は特定のハードウエアや，システムライブラリやOSなどに手を入れることを前提としない汎用性の高いシステムであることが特長である．本報告は，一般ユーザがユーザレベルソフトウエア（移植性のあるDLMライブラリとコンパイラ）のみで，容易に分散大容量メモリを手にできることを示した．

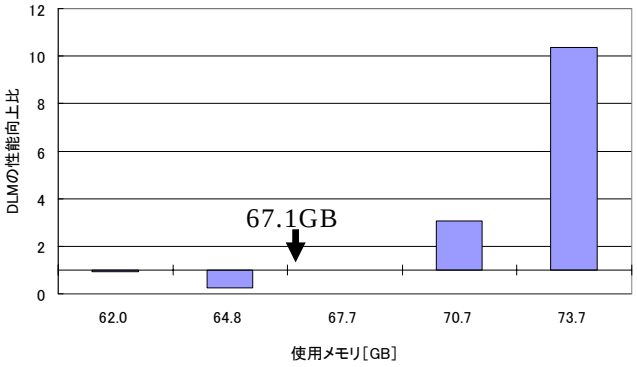

図９　matvの速度向上比(10GbEクラスタ)

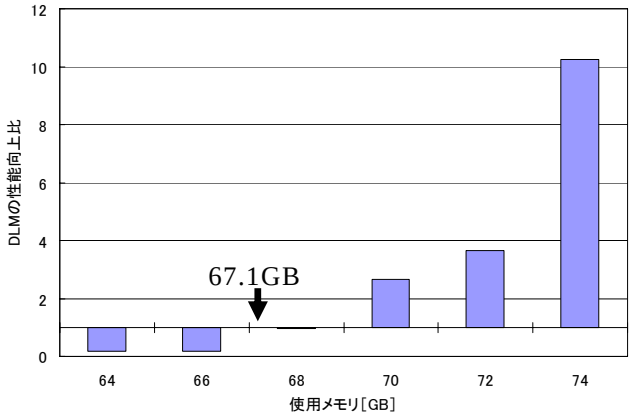

図１０　test0の速度向上比(10GbEクラスタ)

6. 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] L.Iftode, K. Li, K. Petersen, “Memory Server for Multicomputers, ” Proc. 38th IEEE Inter. Computer Conference (Compcon93), pp.534-547, 1993.

[2] 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，”情報処理学会研究報告，2007-HPC-111(21)，pp.121-126, Aug. 2007.

[3] 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，”信学会研究報告，CPSY Vol.106,No.287, pp.7-12,Oct.2006.