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

緑川 博子† 黒川 原佳‡ 姫野 龍太郎§

64bitOS の普及により，飛躍的に大きなアドレス空間が利用可能となった．筆者らは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．この結果，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 5～10 倍以上の性能が得られた．本報告では，ローカル物理メモリの代替として DLM を利用した際の初期性能を調査し，大容量データを用いる応用での DLM 利用に可能性があることを示した．

DLM: Large Memory System using Remote Memory and Compiler

HIROKO MIDORIKAWA† , MOTOYOSHI KUROKAWA‡ and RYUTARO HIMENO‡

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 network-connected computers. Initial performance evaluation shows that DLM programs on cluster with 1Gb/10Gb Ethernet perform more than 5 and 10 times faster than programs using local swap file. It also shows the DLM can be a possible alternative to one high-cost large memory computer for applications with large data The advantages of DLM are not limited in performance but also in easy availability and high portability, because it only uses user-level software and it needs no special hardware.

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通信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なども一つの原因になっている．

ネットワーク上の遠隔メモリをページングに用いるシステムを設計する上で，OS kernel レベルで行うか，ユーザレベルで行うかという２つの選択肢がある．ユーザに完全に透過的である kernel レベルで,遠隔メモリを一つのメモリ階層の一部として組み込むというのが一つの理想像であるが，それには現状のハードディスク特性を前提として設計された OS スワップ機構に，遠隔メモリ用のチューニングを施し，メモリ管理を含めた kernel の再設計が望ましい．しかし現段階では，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[5]-[9]．

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[3][4]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]．ライブラリ実装方式は，移植性が高く，一般ユーザにも導入が容易という利点があるが，前述の他の実装と違い，一般には指定ライブラリを用いてプログラムを書き直す必要があり，ユーザに完全に透過的に遠隔メモリを利用できないという欠点があった．しかし，我々は独自のコンパイラとランタイムシステムを提供することにより，従来の逐次プログラムを大幅に変更することなく，ユーザには透過的に遠隔メモリを利用できる環境を構築した．

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

† 成蹊大学 理工学部 情報科学科 Department of Computer and Information Science, Seikei University
‡ 理化学研究所 情報基盤センターAdvance Center for Computing and Communication, RIKEN
§ 理化学研究所 次世代計算科学研究開発プログラム Research Program for Computational Science, RIKEN

リ利用は，物理メモリ枯渇という緊急状況（kernel によるスワップ処理が発動された状態）で行われるのではなく，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，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．また kernel スワップ機構とは独立であるため kernel 自体の 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指示することも可能である．

1 GbE 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，DLM では 5〜10 倍以上の性能向上が得られた[1][2]．本稿ではさらに，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，主に 10GbE 結合のクラスタ上でいくつかのプログラムで性能を計測したので報告する．

2. DLM ユーザインターフェースと DLM コンパイラ

2.1 dlm 静的宣言と dlm 動的割り当て

DLM を利用するには，逐次 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．下に示す例文のように，静的または動的割り付けを dlm 記述に変更する．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（kernel swap 起動時以外）．また dlm 指定されたデータであっても，ローカル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いくので，遠隔メモリにデータ展開や swap するのは，ローカルメモリ使用量が指定サイズを越えたときだけである．

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

2.2 DLM コンパイラ dlmc

DLM コンパイラは，ユーザには透過的に dlm ライブラリ(dlm_init()，dlm_exit())を挿入し，逐次 C プログラム（図１上）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（図１下）に自動変換する．さらに，ユーザにより dlm 指定された静的データ宣言は，DLM ランタイムシステムにおける動的データ割付方式に合わせ，動的割り当てとポインタ表現に置き換える．

```
#define MAX 1000
dlm int a[MAX];      //  dlm 宣言　唯一の変更箇所

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);  //dlm データをポインタ記述に自動変換

int main(int argc, char *argv[])
{
   int i;
   dlm_init(argc, argv);  //  メモリサーバプロセス遠隔起動,
                          //  通信スレッド生成,  各種初期化
   if(MYPID == 0){  //ローカルホスト（計算スレッド）の処理
   __dlm_dim[0] = 1000;
     __dlm_dim[1] = -1;
     __dlm_div[0] = -1;  //  dlm データの動的割付
     __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();  //メモリサーバプロセスの終了,各種終了処理
}
```

図１　DLM コンパイラによるプログラムの変換例

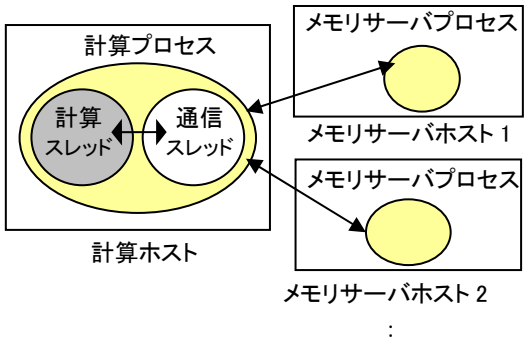
3. DLM ランタイムシステムの構成

3.1 DLM システムの起動と DLM ページ表

DLM システムは図２のような構成であるが，システム起動をユーザに意識させない．(a)のようにユーザがプログラム実行時に DLM 設定ファイル（b）と用いるホスト数を指定するだけで， 自動的に指定数のリモートホストにメモリサーバプロセスが起動され，ローカルホストにメモリサーバプロセスと通信するための通信スレッドが生成される．ユーザのプログラムコードはユーザが起動したプロセス（計算スレッド）で計算される．計算プロセスとメモリサーバプロセス間でソケットを確立させ，DLM 設定ファイルにある利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．ユーザプログラム終了時には，自動的に遠隔プロセスやソケットも終了処理される．

(a)　コマンド例

User_program　−−　–n 3　–f　hostfile



(b)　DLM 設定ファイル　hostfile

```
calhost      32768 //32GB  計算ローカルホスト
memhost1 32768 //32GB  メモリサーバホスト 1
memhost2 65536 //64GB  メモリサーバホスト 2
             :
```

図２ DLM システムの構成

3.2 ページ割り当てと swap 処理要求

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，計算ノード内のローカルメモリへの割り付けを最優先とし，DLM 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 out ページを選び unmap し，該当メモリサーバとの間で要求ページと swap し，新しいページを mmap する．現実装では，手の込んだページ置き換え方式を実装するためにページアクセス時の情報収集オーバヘッドが増大するのを避けるため，swap ページの選択は極力単純化し，ラウンドロビン方式で順に候補を選択している．

4. ローカルメモリ使用プログラムと DLM の性能比較

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 クラスタ上で評価した．用いたのは表１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）である．ノード物理メモリは 64GiB（67.1GB）で，swap 領域を 10GB 持つ．

4.1.1 データアクセス高負荷プログラムにおける性能

ここで用いたのは，8G 個要素の整数配列(32GB) の①初回連続アクセス,②1 ページ(4KB)毎離散アクセス,③再度の連続アクセスを含む図３のプログラム(test1)である．データアクセスのみで計算をほとんど含まないので，一般の応用プログラムに比べ，データアクセス負荷が高く，かつ大容量データを全体にスキャンするため，同じデータが再アクセスされる時間的局所性も乏しいプログラムである．DLM システムではページ prefetch などの連続ページアクセスに有利な処理を行っていないので（ただし DLM ページサイズ内のデータアクセスには，プリフェッチと同等の効果を生んでいる），連続アドレスの DLM ページへのアクセスであろうと，ランダムな DLM ページへのアクセスであろうと，保持ページ以外のアドレスにアクセスがあれば，新しい DLM ページを swap in するという処理上の違いはない．したがって，このテストプログラムの逐次アドレスアクセスが DLM に特に有利になることはなく，むしろ，あるとすれば，ハードディスク上に展開される通常の swap ファイルへのアクセスには，一方向走査なのでシークタイムを減少でき，有利になると考えられる．ここで注目すべきなのはアドレス連続性ではなく，メモリアクセス数に対する swap 回数, swap 頻度である．もし計算がなくメモリアクセスがデータ全体に均一に起こるとしたら，全体のデータに占めるローカルメモリと遠隔メモリの割合によりどの程度の性能低下になるのかを示している．ページ置き換えは単純な割付ページ順のラウンドロビン方式をとっているが，このプ

表１　10GbEther クラスタ　(CSLM)

Cluster	HP DL585 G2 x 5 Nodes
Node CPU	DualCore AMD Opteron(8220SE) 2.8GHz x 4 (8Cores)
Node Memory	64GiByte(67.1GB)
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, HP 431958-B21 (TransRate 300MBps, seektime 4(Ave)8(max)ms)

```
// test1.c dlm data size 32GB = 8000M * sizeof(int) (4B)
#define N  ( (long int) 8000000000)
int main(int argc,char *argv[])
{   int *array;
  unsigned long int i,j;
  array = ( int * )dlm_alloc( sizeof( int) * N );
  for(i=0;i<N;i++)   array[i]=i;   // ①1st sequential access
  for(i=0;i<N;i+=1024) array[i]=-1;   // ②  per page access
  for(j=0;j<N;j++) array[j]=j;   // ③  2nd sequential access
  return 0;
}
```

図３　DLM 逐次アクセスプログラム test1.c

ログラムの場合は LRU と同じ効果を持つ．32GB の dlm データのうち，ローカルメモリと遠隔メモリの割合を変化させて，すべてがローカルメモリ（通常）の場合との速度比を計測した．ここでは DLM ページサイズを 32KB にした場合を示す．

図４，５は①～③各処理における遠隔メモリとの swap 回数，swap 頻度（１swap 当りの平均メモリアクセス数）を示す．①～③はいずれもデータ領域先頭からの書き込みで，①はローカルメモリアクセスに続き，遠隔メモリアクセスが起こり，図４のように全体に占める遠隔メモリの割合に比例して遠隔ページとの swap 回数が増える．②は OS のページ単位（4KB）に１回の書き込みを行う．最初に割り当てられた DLM ページから swap ページに選択されるアルゴリズム上，DLM ページの連続 swap が起きる．したがって図５のように遠隔メモリ使用割合によらず一定回数(DLM ページが 32KB ならば８回書き込みに１回)の swap が起こる．この時が DLM システム利用上，最悪（DLM ページ 4KB のとき）に近い性能状況となる．③も②と同数の swap が起こるが，③は連続書き込みなのでアクセス数に対する swap 頻度は少なくなる．

図６は DLM ページサイズ 32KB，TCP と UDP における①部分のみの速度低下を示す．これによると全メモリアクセス中 0.5%～67%が遠隔メモリアクセスの場合，通常プログラムに比べ 95%～27%程度速度低下を起こす．また UDP と TCP の速度差は少ない．今回 Myrinet-10G の 10GEthernet プロトコルを使用したがメーカ提示の性能特性からも，本クラスタの実環境で測定したネットワーク性能からも TCP が UDP と同等，時には TCP が若干高速であることがわかった．このため， DLM はシステム構築時に UDP，TCP，SDP のプロトコルが選択可能であるが，通常は TCP を使用することとした．

図７は TCP で DLM ページサイズを変化させた場合の①の性能を示す．１回の転送サイズは増えるが，ページサイズが大きくなるほど swap 回数が減り，性能は高くなる．このプログラムは連続アクセスなので DLM ページ内アクセスの prefetch 効果が大きくなるためで，１回のページ転送時間が増加しても転送回数が減るほうが効果のあることがわかる．実際の応用プログラムでは必ずしも連続アドレスアクセスではないため，１回の DLM ページ転送中に中断される計算スレッドのオーバヘッドと DLM サイズとの兼ね合いが生じる．

計測は連続アクセスで行ったので，図６，７の横軸は単にサイズ比というより，プログラム中の全メモリアクセスに占める遠隔メモリアクセスの割合と捉えてもよく，応用における性能を推測する目安となる．DLM の性能はキャッシュと同様にメモリアクセス局所性によるが，図６(DLM ページサイズ 32KB)の場合，このようなデータアクセスが高負荷なプログラムにおいても，

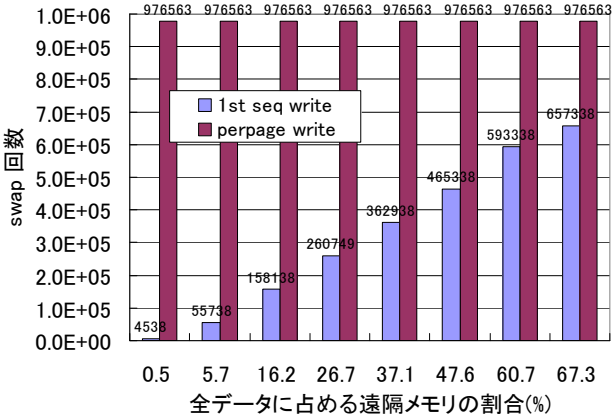


図４　処理①②③の swap 回数（②③は同数）

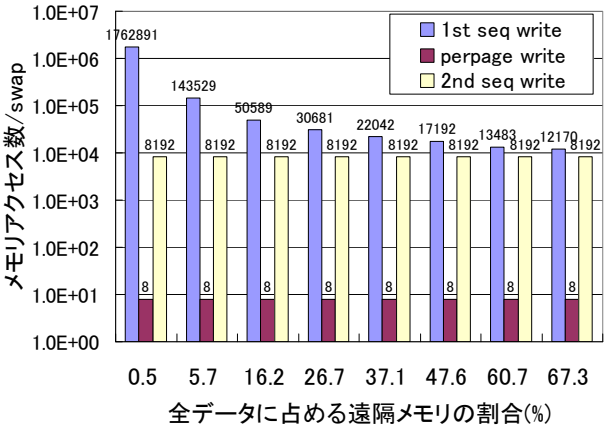


図５　処理①②③の 1swap 当りメモリアクセス数

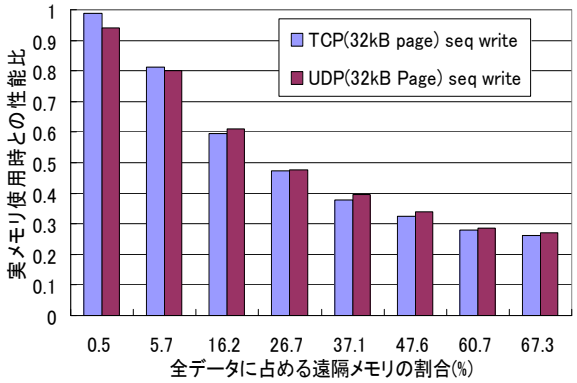


図６　①初回逐次アクセス部分の性能比

遠隔メモリアクセス頻度が全体の 5%以下ならば性能は 80%以上，20%程度ならば性能は半分程度と考えられる．

一方で，②③は 32GB の全データアクセス中，計算がなく，最初から最後までデータアクセス毎に swap を起こす（すなわち使用データのほとんどが遠隔メモリアクセスであるような），アドレス局所性や時間局所性がほとんどない状況に対応する．図８に示すように 2～16

アクセス毎に swap を起こす②では 0.1～0.7%， 2048～16384 アクセス毎に swap を起こす③では 3.7～13.2%にまで性能が低下する．

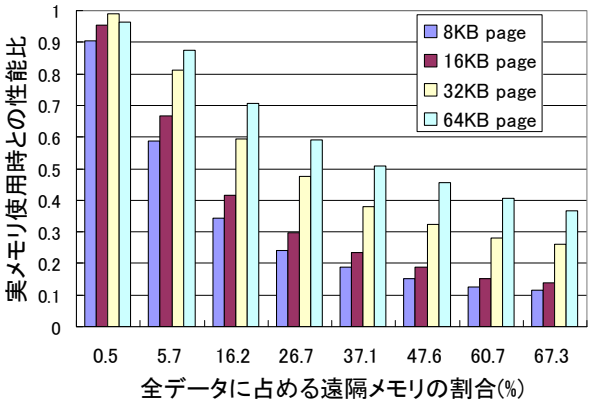


図７　①における DLM ページサイズ（TCP）の影響

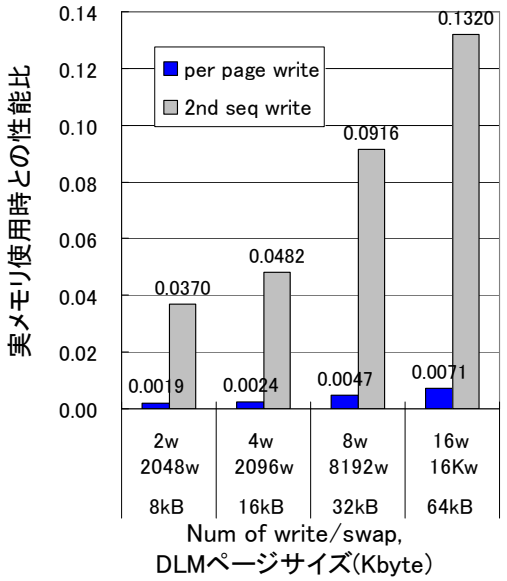


図８　②③部分の速度性能比

4.1.2 swap における DLM ページサイズの影響

１回のメモリアクセスに１回の swap が起きた場合における DLM ページサイズの影響を調べた．メモリアクセス毎に swap を引き起こすように，DLM ページサイズ（４KB～1024KB）以上のアドレス間隔（2MB）で遠隔メモリにある整数配列部分に 1000 回の書き込み（swap）を行うプログラムを作成し，この書き込み部分時間から１回当りの平均書き込み時間を計算したものが，表２である．参考のため，[1]で用いた１GbEthernet クラスタにおける同様のデータ（100 回平均）も示す．これはユーザプログラムレベルの計測なので，ネットワーク通信性能，OS の sigsegv シグナル，map/unmap などの処理，DLM の割り込みハンドラ，ソケット通信，通信スレッドの処理などすべてを含む時間である．右コラムは各クラスタでの DLM ページ４KB の時間を１とした場合の相対値であるが，DLM ページサイズが大きくなると，特に 1GbEhernet では書き込み時間が急速に増し，前節②③のようなアクセスの度に頻繁な swap が起こる応用ではオーバヘッドが非常に高くなる可能性がある．

表２ swap を伴うデータ書き込み平均所要時間

DLM Page Size(KB)	Write Time(usec)		Relative Time	
	1GbE	10GbE	1GbE	10GbE
1024	16495	3106	52.96	29.89
512	7571	1461	24.31	14.06
256	3179	726	10.21	6.99
128	1505	395	4.83	3.81
64	885	245	2.84	2.36
32	559	182	1.80	1.76
16	437	171	1.40	1.65
8	369	116	1.18	1.12
4	311	103	1.00	1.00

4.1.3 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，NPB の C 逐次プログラムを用いて調べた．NPB2.3-omni-C の中から(OpenMP の pragma は無効)，FT，IS，CG の３種（クラスＢ）を用いた結果を示す．各プログラムをローカルメモリだけで実行した時間に比べ，主要データを dlm 宣言して遠隔メモリを用いた場合に，どの程度実行時間が増大したかの倍率を図９，図 10，図 11 に示す．横軸は，DLM ページサイズで４KB～1024KB まで変化させた場合を示している．

いずれの場合にも，DLM ページサイズを大きくするほど実行時間が短く，全体の swap 回数も減少し，DLM ページサイズを中程度（32KB～16KB など）にするより，よい性能が得られた．少なくとも用いた３種のプログラムにおいては，アクセス局所性があり大きな DLM ページを用いたほうが有利であることがわかる．今後広いアドレス空間をアクセスすることを考えると，ページサイズをこれまでのように４KB という小さいままで用いることは，ページ表サイズの増大を招きオーバヘッドを増やす点でも望ましくない．

FT.B は，512x256x256 の 3 次元複素数配列３つと整数配列１つで 1.7GB 程度のメモリを使う．NPB の他のプログラムよりも計算量に比べデータ容量を多く必要とし，３次元の異なる方向へのデータアクセスを毎回全領域に対して行う繰り返しがあるため，この３種の中で最も swap 回数，頻度が高い．ローカルメモリ/遠隔メモリの比率の高低によって性能低下の度合いに大きな差はない．離散とはいえ一定間隔一定方向へのアクセスが多いためか，prefetch 効果による swap 回数低減に

効果のある DLM ページの大きさが影響する．

IS.B は，2^{25}のサイズの整数配列３つで 384MB 程度のメモリを用い比較的サイズは小さいが，アクセスの特性からかローカルメモリ／遠隔メモリ比率に応じて性能が大きく低下する．また DLM サイズの影響もローカルメモリ比率が低い場合には大きく影響する．

CG.B は他の２つに比べて特異で，14 種の配列で約 510MB を使用しているが，ローカルメモリに 30～40%程度のデータがあれば，DLM ページサイズに拠らず性能低下はほとんどない．これは計算量に比べ swap 回数が非常に少なく swap 頻度が低いためである．ただし，遠隔メモリ量の割合が一定量を超えると（ローカルメモ 100MB 以下）と，急速に性能が落ちる．CG は行列アクセスの際にインデックス格納配列要素を用いて行列に間接アクセスする処理が多く，どちらかが必ず遠隔メモリにあるような状態を引き起こすと，swap 回数が急速に増すためではないかと考えられる．

5. おわりに

今後，様々なアクセスパターンと計算を含むプログラム，動的メモリ割付や解放を頻繁に行うプログラムなどを含むについても性能調査し，ページ置き換えアルゴリズムにも工夫ができないか検討する予定である．しかし現在の初期実装でも，従来のローカルディスクを用いる swap デーモン処理に比べ，遠隔メモリを利用する方式は格段に高速なのは事実で，紙面の都合で割愛したが 1 GB ノードメモリを搭載する 1 GbEthernet クラスタ[1]においてでさえ，上記 FT.B を通常実行すると 28 時間を要したが，DLM を用いると約 40 分で実行することができた．メモリアクセス局所性が一定度以上あれば，搭載メモリを超える容量のデータを扱う処理にも DLM を利用できる可能性を示した．

参 考 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", 電子情報通信学会 CPSY 研究会報告,信学技報 Vol.102, No.398, pp.29-34, Dec.2007

[2] 緑川,小山,黒川,姫野, "分散大容量メモリシステム DLM の初期性能評価", HPCS2008, p.66, Jan.2008

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

[4] 緑川，飯塚：″ユーザレベル・ソフトウエア分散共有メモリ SMS の設計と実装″，情処論文誌,Vol.42, No.SIG9(HPS 3), pp.170-190, Aug.2001

[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, Sept. 2005

[6] Pavel Mache, Linux Network Block Device, (1997) http://www.xss.co.at/linux/NBD/

[7] 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.

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

[9] 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 研究会報告，信学技報告 Vol.106 No.287, Oct.2006

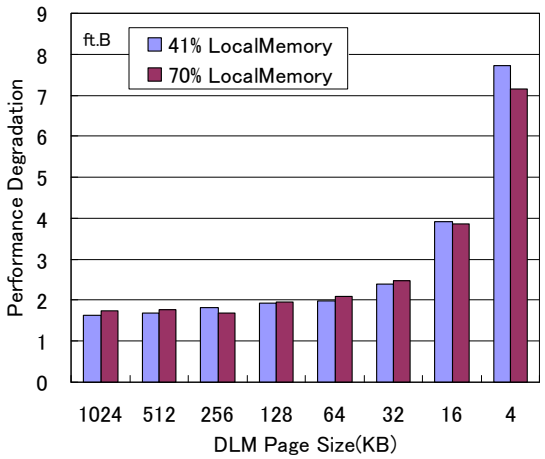


図 9　FT.B　通常時と比較した実行時間の増大率

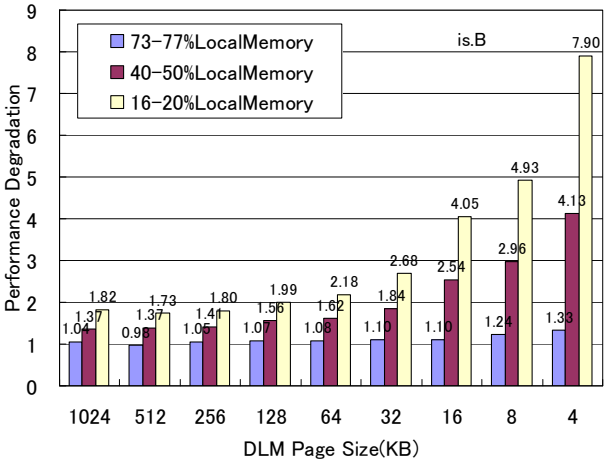


図 10　IS.B　通常時と比較した実行時間の増大率

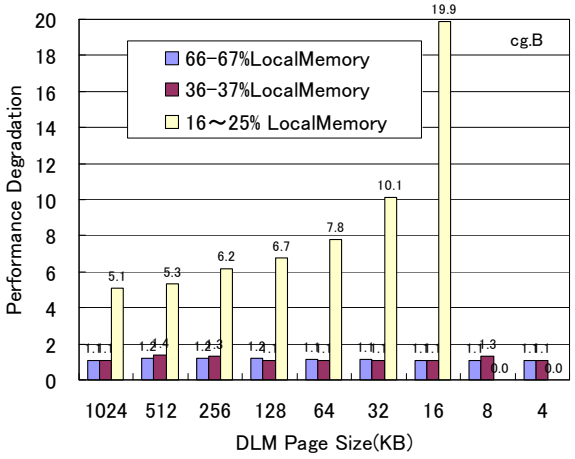


図 11　CG.B　通常時と比較した実行時間の増大率