2H-04

遠隔メモリページングにおけるローカルデータへの get/put 機構による高速データ転送の実装

大浦　陽†　　緑川　博子‡　甲斐　宗徳†3

成蹊大学†

1.はじめに

1.1 背景

大規模科学技術計算などの大量のメモリを用いる応用プログラムがあるが，ハードウェア制限や消費電力などの問題から 1 つのコンピューターに搭載できる主メモリの容量には限界がある．そこで高速ネットワークを用いて複数のコンピューターをクラスタ上で繋ぎ，遠隔メモリページングを行うことで，仮想的な大容量メモリを提供する分散大容量メモリシステム(DLM)を開発してきた．

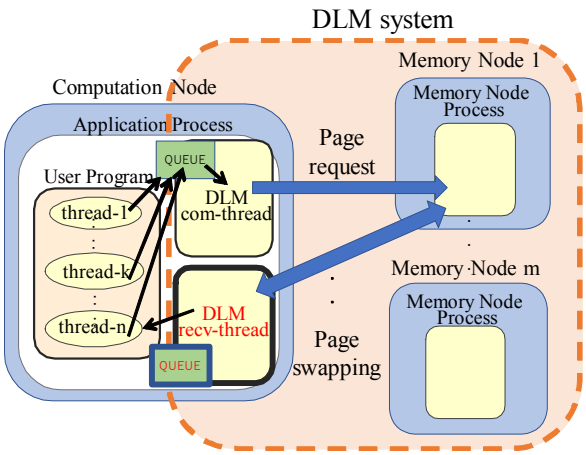

図 1.1 DLM の構成

DLM[1]は，主メモリサイズを超えるサイズのデータを扱う Out-Of^Core 処理を目的としている．複数ノードに分散配置された大域データのどこにでもアクセスできるが，分散共有メモリ DSM のように、複数ノードによる並列処理は行わない．単一システム上の共有メモリモデルを基に作成されたプログラムやアルゴリズムが変更なしに移行できるようにするためである．

DLM は 1 台の計算ノードと複数のメモリノードから構成され，計算ノードでユーザープログラムの実行し，メモリノードは計算ノードに収まらないデータと計算ノードのデータのコピーを格納する．DLM においてデータエリアを確保する場合，計算ノードメモリのみに割り付けられるデータをローカルデータ，計算ノードとメモリノードのメモリに割り付けられるデータを DLM データと呼び，区別できる．

1.2 目的

DLM は Segv シグナルを利用して，DLM データに対しどのようなデータアクセスでも対応できる実装だが，1 枚ずつページ交換を行うため，必ずしも効率的ではない場合がある．本報告では，計算ノードとメモリノード間のデータ転送効率を向上させるため，ローカルデータ領域と DLM データ領域のデータコピー機能(get/put)を新たに DLM に実装し，二つのベンチマークプログラムを用いて初期評価実験を行った．

2.get/put 機構

2.1 get/put 機構

GA[2]などの PGAS ランタイムにはユーザーがノード間のデータ送受信を明示して，ローカル領域とグローバル領域のデータを操作する機能(get/put)がある．ユーザーが指定した DLM データをローカルデータへとコピーする get 機構，ユーザーが指定したローカルデータを DLM データへコピーする put 機構を DLM に取り入れた．

get/put 機構の利点として次の三点がある．

1. 計算中のページ取得時の Segv 発生を抑制
2. ページ単位ではなく連続領域の一括転送
3. ローカルデータ使用時に効率的メモリ利用が可能

DLM では遠隔ページにプログラムがアクセスすると Segv が発生し，他スレッドを一時停止させメモリノードから該当ページを受け取り，代わりに計算ノードのページを送信する．get/put により，計算前に必要なデータをあらかじめ get（ローカルデータへのコピー）することで，計算中の Segv を抑制し，各スレッドの計算が中断されず性能向上が見込める．さらにユーザーが指定した連続領域の一括転送により，ページ単位の転送に比べデータ転送回数を低減できる，DLM データから直接ローカルデータへコピーすることで，計算ノードに保持する DLM データ量を減らし，ローカルデータ領域を増大させることができ，ローカルメモリの有効利用が可能である．

2.2 get_array/put_array 機構

get/put は連続アドレス領域を対象とするが，高性能計算等では，多次元配列を扱うことが多い．そこで多次元配列から部分配列をコピーするための get_array/put_array を実装した．

Implementation of high-speed data transfer by get / put mechanism to local data in remote memory paging

†Hikari Oura ・Seikei University

‡Midorikawa Hiroko ・ Seikei University

†3 Kai Munenori ・ Seikei University

3.初期評価実験

get/put と get_array/put_array の評価をベクトル和と 3 次元データに対する 7 点ステンシル計算[3]を用いて初期評価実験を行った．実行環境は表 1 に示す．各実験のパラメータは表 2 に示す．

3.1 ベクトル和

一次元配列のベクトル和を用いた初期評価実験を行った．DLM データとして 128GiB の配列を 3 つ確保し，ローカルデータとして 32GiB の配列を 3 つ確保した．DLM データからローカルデータへのデータコピーを次の二通りを用いて実行した．

方式 1.　get/put を用いる （事前/後一括データ転送）

方式 2.　memcpy を用いる（segv によるページ転送）

図 1 はベクトル和における方式 1 と方式 2 の違いを簡単なコードで表したものである．演算前と演算後に大配列から小配列へデータコピーを行い，メモリアクセスローカリティを高める．

図 2 はベクトル和の実行時間と各処理の内訳を示す．

CPU	Intel® Xeon® CPU E5-2687W v3 @ 3.10GHz 2CPU × 10core/node
Memory	128GB/node
Cache	L2 256KiB L3 25MiB
Network	Infiniband FDR
OS	CentOS 7.1.1503 (Core) Linux 3.19.5
Compiler	gcc version 4.8.3
MPILib	MVAPICH2 version 2.0.1

表 1 実行環境

	ベクトル和	7点ステンシル
総配列量	128 GiB * 3	64~256GiB
ローカルデータ(ブロック)サイズ	1~32GiB * 3	20GiB * 2 60GiB * 2
使用ノード台数	4	4
計算ノード主メモリ量	117GiB	117GiB
メモノード主メモリ量	117GiB	117GiB
スレッド数	1~16	1~16
計算ノードメモリ量/全データ量	15%	100%~45%
ページサイズ	8MiB	8MiB

表 2 実験パラメータ

```
#define N BigNumber
#define K SmallNumber
int* Aarray = (int*)dlm_alloc(sizeof(int)*N);  //128GiB*3
int* Asubarray=(int*)malloc(sizeof(int)*N/K);  //32G
//B,Cも同様に確保する
for(int j =0 ; j < N; j += N/K){
        dlm_get(Asubarray,Aarray[j],N/K); or memcpy(Asubarray,Aarray[j],N/K);
        dlm_get(Bsubarray,Barray[j],N/K); or memcpy(Bsubarray,Barray[j],N/K);
        #pragma omp for
        for(int i=0; i < N/K;i++)
                Csubarray[i] = Asubarray[i] + Bsubarray[i] ;
        dlm_put(Carray[j],Csubarray,N/K); or memcpy(Carray[j],Csubarray,N/K);
}
```

図 1 ベクトル和における方式 1 と方式 2 の違い

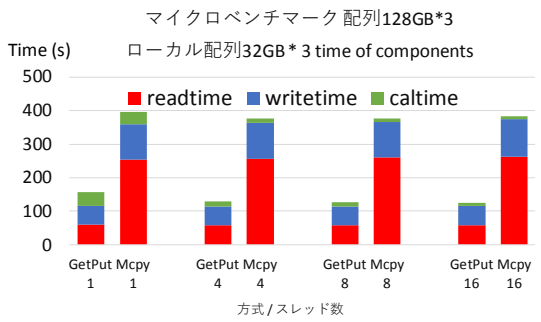

図 2 ベクトル和の実行時間とその内訳

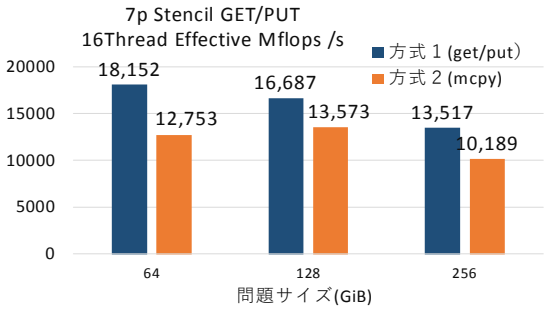

図 3 7 点ステンシル計算の Effective Mflops

左が方式 1 を示し，右が方式 2 を示す．方式 1(get/put)は方式 2(memcopy)の約 30%の実行時間である．

3.2　7 点ステンシル計算

ステンシル計算は基本的な格子計算の一つである．ここでは，時間・空間ブロッキング手法を用いている．3 次元格子データ配列全体を DLM データとして，ブロッキングを用いて計算を行う部分配列をローカルデータに確保し計算する．DLM データからローカルデータへのデータコピーを次の二通で行った．

方式 1.　get_array/put_array（事前/後一括データ転送）

方式 2.　配列データ計算時アクセス（segv による）

図 3 は両方式の性能を示す．問題サイズ 256GiB の時，方式 1 は方式 2 と比べ 1.3 倍の性能で実行できた．

4.終わりに

本報告では，計算ノードとメモリノード間のデータ転送効率を向上させるために，ローカルデータエリアと DLM データエリアのデータ送受信機能を DLM に実装した．2 つのベンチマークプログラムを用いて初期評価実験を行い，ベクトル和では 67%，7 点ステンシルでは 30%の性能向上が見られた．

5.参考文献

[1] H.Midorikawa, et.al,"Using a cluster as a memory resource : a fast and large virtual memory on MPI", IEEE Cluster2009, pp. 1-10

[2] GlobalArrays Webpage.http://hpc.pnl.gov/globalarrays

[3] H Midorikawa, et.al.: "An evaluation of the potential of Flash SSD as large and slow memory for Stencil computations", IEEE HPCS2014, PP.268-277