B-005

マルチノードマルチコア向け分散共有メモリにおけるデータ分散配置 API の導入
The Introduction of Data Distribution Interface in Distributed Shared Memory for Multi-node and Multi-Core Processing

白澤 卓磨† 緑川 博子† 甲斐 宗徳†
Takuma Shirasawa Hiroko Midorikawa Munenori Kai

1. はじめに

近年，HPC(High Performance Computing)分野において，大規模な問題を解くために，マルチコア・マルチノードを用いた並列処理で高速化させることが求められ，現在では，MPI(Message Passing Interface)と，OpenMP などのスレッドプログラミングを組み合わせたプログラミングが主流となっている．しかし，MPI はデータをローカルビューで扱うので，ユーザプログラムの生産性が低くなってしまう問題がある．そこで，グローバルビューを提供する分散共有メモリ SDSM(Software Distributed Shared Memory)が 1990 年代から開発されてきた[1][2]．SDSM とは，分散メモリ環境下において，計算ノード間のメモリをあたかも共有メモリのごとく見せる技術である．最近では高速通信(InfiniBand+MPI)やマルチコアなどを用いた，新しい SDSM も提案されており，本研究でも，マルチノードマルチコア向け SDSM である，M-SMS[3]を試作している．

SDSM は，全てがフラットな共有メモリであるかのような処理を行うと性能が劣化するので，データアクセス局所性を高めるための PGAS(Partitioned Global Address Space)モデルが提案されている．M-SMS で実装されているデータ分散配置機能(以下，分散マッピングと呼ぶ)により，データアクセス局所性を考慮したプログラムと組み合わせることで性能向上が期待できる．本研究では，ノード間の共有データ配置をよりわかりやすく記述するために，MpC 言語[4]の分散マッピング API を導入した．また，これを用いて，分散マッピングとデータアクセス局所性を考慮したプログラムを作成し，性能を評価した．

1.1 ページベース分散共有メモリの概要

SDSM は，OS のメモリ保護機構を用いて，ノード外のメモリとノード内のメモリのデータの位置をシステム側が自動的に識別し，ユーザ指定のページサイズでデータを通信して取得・管理することで実現している．

ノード間で共有されているデータはページ単位で管理をし，各ノードが管理しているページを決めている．そのため，管理外のページにアクセスするときは，そのページを管理しているノードからページをキャッシュすることで，全てのデータへアクセスすることが可能になっている．

1.2 M-SMS による並列処理

M-SMS では，グローバルビューデータを提供し，既存の OpenMP や Pthread のようなマルチスレッドプログラムによるノード内並列と，SPMD(Single Program Multiple Data)記述によるノード間並列の二つの並列プログラミングモデルを提供している．

†成蹊大学理工学研究科理工学専攻 Graduate School of Science and Technology, Seikei University

また，sms_barrier 関数により，各ノード間の実行同期と共有データの一貫性を取っている．

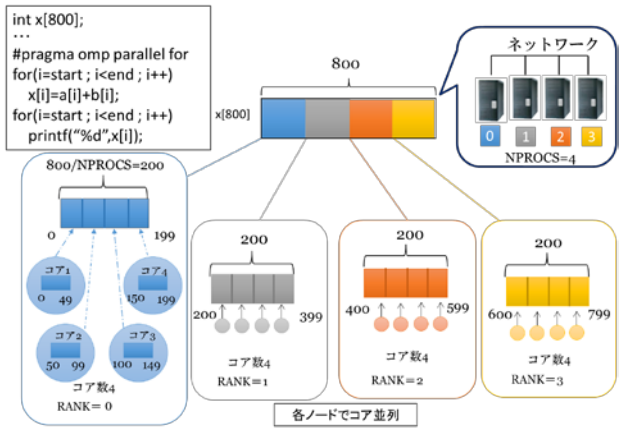

図 1 M-SMS による並列処理イメージ

2. 分散マッピング機能

分散マッピング機能は，全ノードが認識し共有することのできるグローバルビューのデータを，各ノードに分散して配置する機能のことである．データアクセス局所性を向上させるためには，全ノードが認識している共有データへのアクセスを，できるだけ自ノードで管理している共有データで済ませることが重要である．

2.1 共有データの分散配置

共有データをどのように配置するのかは，ユーザがプログラム上で指定する．図 2 は，2 次元配列と 3 次元配列において共有データを 4 ノードに分散配置した例の模式図である．

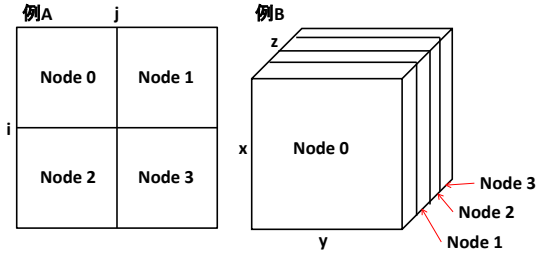

図 2 共有データの分散配置例

M-SMS では，sms_mapalloc 関数を用いることで分散マッピングを使用することができる．本研究では，後述する MpC 言語を導入することで，MpC コンパイラ[4]が分散マッピング API を sms_mapalloc 関数に変換し，M-SMS の分散マッピング機能を利用している．

3. MpC 言語

PGAS モデルの代表例としては，Chapel[5]，X10[6]，XcalableMP[7]などがあるが，言語によってはデータ配置や処理の柔軟性に制限があることや，ノード間で共通のデータにアクセスする範囲が限定されているなどの制約があることがある．

本研究で使用している MpC 言語は，SMS[2]に基づいて開発された言語であり，この言語を M-SMS に対応させることで分散マッピング API を利用可能とし，各ノードへの共有データ割付の記述をより簡単に行えるようにした．

3.1 分散マッピング API

図 3 では，MpC 言語における分散マッピング API の使用形式と，図 2 の例のような分散マッピングを行う際の記述例を示す．

shared 型名 変数名[sm]…[s1]…[s0]::[dm]…[d1]…[d0](st,n)
変数名[sm]…[s1]…[s0] → 各次元の要素数
[dm]…[d1]…[d0] → 各次元の分割数（省略時は1）
st → 割り付け開始ノード番号（省略時は0か任意）
n → 割り付けノード数（省略時は全使用ノード数）
記述例
shared int arrayA[i][j]::[2][2](0,4) //例A
shared int arrayB[z][y][x]::[4][1][1](0,4) //例B

図 3　分散マッピング API の使用形式と記述例

C 言語における通常の配列表記に，共有データ型として頭に shared を付け，各次元の分割数，割り付け開始のノード番号，割り付けノード数を指定する．これにより，各ノードが共有するデータがどのように分割され管理されるのかを容易に記述できる．図 4 は MpC 言語で書いたプログラムの記述例である．

```
shared double A[N][N]::[NPROCS][1](0,NPROCS);
・・・
mpc_init(&argc,&argv);
・・・
#pragma omp parallel for private(j) num_threads(8)
        for(i=MYPID*(N/NPROCS);i<(MYPID+1)*(N/NPROCS);i++)
          for(j=0;j<N;j++)
                  A[i][j]=・・・;//計算部分
・・・
sms_barrier();
```

図 4　MpC 言語プログラムの記述例

4. 性能評価

分散マッピング API を利用し，データアクセス局所性を考慮したプログラムを用いて，共有データの割り付け方法による実行時間の違いを比較した．

4.1 実験方法

共有データを 4 つのノードに分散した割り付け方法と，1 つのノードに集中させて割り付けた方法，2 つの割り付け方法で 3 次元 7 点ステンシル計算を行った．double 型 3 次元配列を対象とし，同様の計算を 4 回繰り返した．また，問題サイズは 128x128x128 と 256x256x256 で実行した．データ割り付けは，図 5 に示す集中方式と分散方式を用いた．

計算部分は 4 ノード 8 スレッドの並列処理で行った．その際，ノード間の並列では 4 ノードで分散したデータへのアクセスをできるだけローカルアクセスにするようにした．

```
shared double a[N][N][N]::[4][1][1](0,4) //4ノード分散
shared double b[N][N][N]::[1][1][1](0,1) //1ノード集中
```

図 5　実験での割り付け方法による API の記述

ただし，ステンシル計算は周りの要素を利用するので，割り付けたデータの境界部分では，自ノードでは管理していないデータを他のノードからキャッシュすることになる．一方で，1 ノード集中割り付けでは計算の際，共有データを割り付けたノードからデータをキャッシュしてから計算をすることになる．なお，管理するデータのページサイズは 1MB とした．表 1 は実行環境である．

表 1　実行環境

CPU	Intel Xeon CPU E5-2687W v3 3.10GHz
Memory	128GB (4nodes)
Network	InfiniBand singleFDR (56Gbps)
OS	CentOS 7.1.1503
Compiler	gcc version 4.8.3

4.2 実験結果

図 6 は各割付方式による実行時間を示す．4 ノード分散割り付けは集中割り付けに比べ，128x128x128 では 8%，256x256x256 では 2%まで実行時間が減っている．分散マッピング機能によりデータアクセス局所性を高めた結果であると言える．

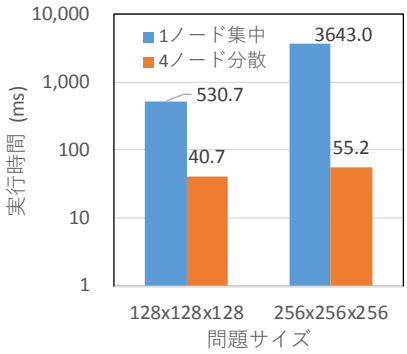

図 6　割り付け方法の違いによる比較

5. おわりに

MpC 言語の分散マッピング API を導入し，共有データを配置する記述がよりわかりやすくなった．また，データアクセス局所性を向上させることで，プログラムの性能が向上した．

参考文献

[1] P.Keleher, et.al "TreadMarks: Distributed shared memory on standard workstations and operating system", Proc. of the 1994 Winter Usenix Conference, pp.115-131, 1994.
[2] 緑川，飯塚："ユーザレベル・ソフトウェア分散共有メモリ SMS の設計と実装",情報処理学会論文誌 Vol.42,No.SIG9, pp.170-190,2001
[3] 緑川,岩井田："マルチスレッド対応型分散共有メモリシステムの設計と実装", HPCS2015, HPCS2015 論文集,（2015,5-19）
[4] 緑川, 飯塚:"メタプロセスモデルに基づくポータブルな並列プログラミングインタフェース MpC",情報処理学会論文誌 Vol.46 No.SIG4, pp.69-85,2005
[5] http://chapel.cray.com/
[6] http://x10-lang.org/
[7] http://www.xcalablemp.org/