Vol.2011-ARC-197 No.36
Vol.2011-HPC-132 No.36
2011/11/29

並列PGASプログラミング言語XcalableMPの入出力機能とLustreファイルシステムでの性能評価

中村朋健[†1]　佐藤三久[†2,†1]

大規模な並列システムにおいて，各ノードにあるデータを効率的に入出力するためには，MPI-IO などの並列入出力を活用する必要がある．並列プログラミング言語 XcalableMP は，そのグローバルビューのプログラミングモデルにおいては，複数のノードに跨る分散配列を定義することができ，その分散に従った典型的なデータ並列プログラミングをサポートしている．このような分散配列にあるデータを効率的に入出力するためには並列入出力機能を活用する必要があり，言語でサポートされている分散情報を用いることにより簡便かつ効率的に入出力をすることができる．XMP-IO は，XcalableMP の仕様の一部として定義されている並列 IO インタフェースである．それに加えて、我々はよく使われる NetCDF ファイルを XcalableMP から並列に入出力するためのライブラリインタフェースを設計した．これらの XcalableMP の並列入出力機能について，並列ファイルシステム Lustre 上で並列入出力を評価し，その有効性を確かめた．

Parallel I/O Facility of PGAS Programming Language XcalableMP and Evaluation of I/O Performance on the Lustre File System

Tomotake NAKAMURA[†1] and Mitsuhisa SATO[†2,†1]

In a large-scale parallel system, parallel I/O libraries such as MPI-IO are required to perform I/O operation between nodes and file systems. XcalableMP is a PGAS parallel programming language, which allows the programmer to define a distributed array over nodes to support typical data parallel programming with global-view programming. In order to perform I/O operations of a distributed array efficiently, the information of a distributed array described in the program can be used for parallel IO operations. XMP-IO is a parallel I/O API defined as a part of the XcalableMP specification. As other parallel I/O interface, we also designed a XMP library interface to a parallel library for netCDF which is a commonly used file format. The experimental results of these parallel IO in XcalableMP shows that these parallel I/O interfaces provide good parallel I/O performance on the Lustre file system with reasonable programming cost.

1. はじめに

データを分散させ各プロセッサで並列に処理するアプリケーションを作成するために，MPI のようなライブラリや並列プログラミング言語（並列言語）[1),3),2),4)] が数多く研究開発されている．大規模な分散メモリの計算機上で並列処理をするためには多くのノードにおいて，データを入出力することになる．計算機の規模が大きくなるに連れて入出力するデータが大きくなるため，特定のノードにデータを集めて入出力することは困難になる．また，このうな環境で動かすアプリケーションでは，これを解決するために，各ノードにあるデータを独立に入出力することが考えらえれるが，膨大な数のノードが同時に同じファイルに入出力する場合，排他制御が必要となり著しく性能が低下してしまう．他の方法としては，ノードごとに別々のファイルを用いる方法があるが，膨大な数のファイルができてしまうなど，管理上の問題が生じる．その解決方法として，MPI においては MPI-IO などの複数のノードからの効率的に並列入出力する機能が提供されている．

WRF(the Weather and Research Forecasting Model)[7)] のような大規模なアプリケーションで逐次の入出力が実行時間のボトルネックになっている[5),6)]．この問題を解決するために，並列言語には並列入出力が必須の機能である．並列言語で並列入出力機能を効率よく使うためには，ハードウェア構成やファイルシステムが並列入出力に対応していなければならない．近年の大規模な計算機は並列に入出力可能なファイルシステムを備えており，並列入出力の性能効率を意識したハードウェア構成となっていることが多い．

並列プログラミング言語 XcalableMP は Partitioned Global Address Space(PGAS) 言語であるが，そのグローバルビューのプログラミングモデルにおいては，複数のノードに跨る分散配列を定義することができ，その分散に従った典型的なデータ並列プログラミングをサポートしている．このような分散配列にあるデータを効率的に入出力するためには並列入

†1 理化学研究所 計算科学研究機構
Advanced Institute for Computational Science, RIKEN
†2 筑波大学 計算科学研究センター
Center for Computational Sciences, University of Tsukuba

Vol.2011-ARC-197 No.36
Vol.2011-HPC-132 No.36
2011/11/29

出力機能を活用する必要があり，言語でサポートされている分散情報を用いることにより簡便かつ効率的に入出力をすることができる．

XcalableMP の入出力機能 (XMP-IO) は，現在，XMP 仕様 WG において，XcalableMP の仕様の一部として議論されている．XMP-IO はローカル I/O，マスタ I/O，グローバル I/O の 3 種類の入出力方法がある．ローカル I/O は各実行ノードがそれぞれのファイルに対して入出力する．マスタ I/O は代表のノード（マスタノード）が 1 つのファイルに対して入出力する．グローバル I/O は各実行ノードが 1 つのファイルに集団的，または，独立に入出力する．

一般に，XMP-IO のグローバル I/O は，大規模なノード構成で，マスタ I/O よりもメモリ使用量を少なく入出力可能である．また，グローバル I/O は並列の入出力が可能であるため，逐次の入出力でボトルネックになるようなアプリケーションに対して効率よく並列に入出力可能な機能である．

科学アプリケーションでよく利用されるファイルフォーマットの一つに NetCDF[8] ファイルがある．NetCDF で扱うデータフォーマットは XMP-IO が扱うファイルフォーマットと異なるため，XMP-IO で NetCDF ファイルを入出力することはできない．XcalabMP から NetCDF ファイルを入出力可能になると XcalableMP で既存の資産を有効に使える．

NetCDF ファイルを並列に入出力可能な Parallel NetCDF API[9),10)] がすでに開発されている．我々は XcalableMP の拡張機能として Parallel NetCDF API を呼び出して NetCDF ファイルを操作するライブラリインタフェースを設計している．

本稿では並列に入出力可能な MPI-IO，Parallel NetCDF，XcalableMP のグローバル I/O そして逐次に入出力したプログラムを並列に入出力可能なファイルシステム Lustre 上で性能評価する．ここで，XcalableMP のグローバル I/O は XMP 仕様 WG 議論されている仕様に基づいて実装，性能評価した．

本稿は 2 章で XcalableMP の入出力機能 (XMP-IO) について現時点での有力な仕様を説明する．4 章で並列入出力と逐次入出力を Lustre 上で性能評価する．3 章で，XcalableMP の拡張機能として設計中の NetCDF ファイルを操作するインタフェースを説明する．5 章で本稿をまとめる．

2. XcalableMP の入出力機能 XMP-IO

XcalableMP の入出力機能 (XMP-IO) にはローカル I/O，マスタ I/O，グローバル I/O の 3 種類の入出力方法がある． 3 種類の入出力方法のデータの流れを図 1，図 2，図 3 に示す．

XMP-IO は 2 つのファイルポインタ（固有ファイルポインタ，共有ファイルポインタ）を用いてファイルからデータを読み込む，または，ファイルへデータを書き出す．ここで，ファイルポインタとはファイルへ読み書きするためのファイルの位置を指すものである．固有ファイルポインタは各ノードがファイルに対してもつファイルポインタであり，共有ファイルポインタはすべての実行ノードがファイルに対して 1 つだけもつファイルポインタである．本章では XMP-IO の各入出力について説明する．

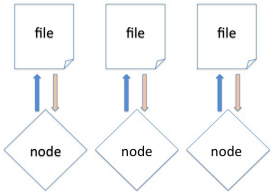

図 1 ローカル I/O

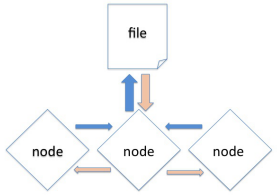

図 2 マスタ I/O

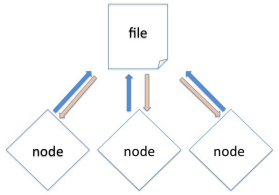

図 3 グローバル I/O

2.1 ローカル I/O

ローカル I/O はベース言語の入出力文や入出力サービス関数をそのまま利用してファイルを操作する方法である（XcalableMP のベース言語は Fortran と C である）．XcalableMP 指示文で指定されていない入出力文や入出力サービス関数はローカル I/O となる．ローカル I/O を用いて各実行ノードが他のノードと独立にファイルへ入出力や接続・接続解除などのファイルを操作することができる．

ローカル I/O のファイルへの接続・接続解除はベース言語の接続・接続解除と同様である．ローカル I/O の出力文で書き出したファイルをローカル I/O の入力文で読み込むことができる．また，ローカル I/O の入力文はベース言語で書き出したファイルを読み込むことができ，ローカル I/O の出力文で書き出したファイルをベース言語で読み込むこともできる．

ローカル I/O 文はローカル変数に加え分散配列の値を入出力することができる．ローカル変数を入力項目としたとき，各ノードでローカル変数の値が読み込まれる．ローカル変数や一般の式を出力項目としたとき，各ノードでそれらを評価した値が書き出される．分散配列の全体配列を入力項目，または，出力項目としたとき，各ノードで確保した配列に読み込

Vol.2011-ARC-197 No.36
Vol.2011-HPC-132 No.36
2011/11/29

まれる，または，配列が書き出される．分散配列の部分配列を入出力項目にすることはできない．

2.2 マスタ I/O

マスタ I/O はベース言語が Fortran のときのみサポートされ，C のときはサポートされない．

マスタ I/O は実行ノード集合が一つのファイルに対して入出力などのファイルを操作する方法である．マスタ I/O を使って実行ノード集合が集団的 (collective) にファイルへ入出力や接続・接続解除などの操作ができる．マスタ I/O は実行ノードを代表する 1 ノード（マスタノード）が固有ファイルポインタを用いてファイルへ操作する．

Syntax 1

```
!$xmp master_io
        io-statement
```

Syntax 2

```
!$xmp master_io begin
        io-statement-list
!$xmp end master_io
```

図 4 マスタ I/O の構文

マスタ I/O にするために OPEN 文，CLOSE 文，READ 文，WRITE 文に指示文をつける．グローバル I/O の指定方法を図 4 に示す．マスタ I/O のファイルへの接続・接続解除はマスタノードが実行する．マスタ I/O 文でファイルからローカル変数に値を読み込んだとき，接続されているすべての実行ノードのそのローカル変数は同じ値になる．ローカル変数や一般の式をマスタ I/O の出力項目としたとき，マスタノードの値がファイルに書き出される．ファイルから分散配列の全体配列に読み込むとき，グローバルビューでの配列要素の並び順でファイルからマスタノードに読み込み，割り当てられたノードの配列へ転送し配列へ設定される．分散配列の全体配列をファイルに書き出すとき，割り当てられたノードの配列からマスタノードに転送し，グローバルビューでの配列要素の並び順に書き出される．ファイルから分散配列の配列要素に読み込むとき，マスタノードで読み込み割り当てられたノードの配列へ転送し配列へ設定される．分散配列の配列要素をファイルに書き出すとき，割り当てられたノードの配列からマスタノードに転送し書き出される．入出力項目に分散配列の部分配列，文字部分列，添字に分散配列の引用を含む配列要素と部分配列と文字部分列，分散配列の引用を含む式，分散配列の引用を含む DO 制御を指定することはできない．

2.3 グローバル I/O

グローバル I/O は実行ノードが一つのファイルに対して入出力などのファイルを操作する方法である．グローバル I/O は集団的（collective）グローバル I/O，排他的（atomic）グローバル I/O，位置指定（direct）グローバル I/O の 3 種類の I/O 方法がある．ベース言語が Fortran のとき，グローバル I/O にするために OPEN 文，CLOSE 文，i READ 文，WRITE 文に指示文をつける．ベース言語が Fortran のときのグローバル I/O の指定方法を図 5 に示す．

Syntax 1

```
!$xmp global_io [atomic / direct]
        io-statement
```

Syntax 2

```
!$xmp global_io [atomic / direct] begin
        io-statement-list
!$xmp end global_io
```

図 5 ベース言語が Fortran のときのグローバル I/O の構文

ベース言語が C のとき，グローバル I/O にするためにサービス関数とライブラリ関数を用いる．ライブラリ関数は以下の型の構造体を引数にすることがある．

- xmp_file_t （ファイルハンドル）
- xmp_array_t （分散配列の配列情報）
- xmp_range_t （部分配列の範囲の情報）

3 種類のグローバル I/O で共通して用いられるファイル操作関数を以下に示す．ここで，関数名の末尾が” _all” である関数は集団実行の関数である．以降で記述するライブラリ関数の詳細につては XcalableMP 仕様書を参照されたい．

Vol.2011-ARC-197 No.36
Vol.2011-HPC-132 No.36
2011/11/29

xmp_fopen_all(const char *fname, const char *amode)
ファイルをオープンする．返り値がファイルハンドルである．

xmp_fclose_all(xmp_file_t *fh)
ファイルをクローズする．

xmp_fseek(xmp_file_t *fh, long long offset, int whence)
固有ファイルポインタの設定を変更する．

xmp_fseek_shared_all(xmp_file_t *fh, long long offset, int whence)
共有ファイルポインタの位置を変更する．

xmp_ftell(xmp_file_t *fh)
固有ファイルポインタの位置を問い合わせる．

xmp_ftell_shared(xmp_file_t *fh)
共有ファイルポインタの位置を問い合わせる．

xmp_file_sync_all(xmp_file_t *fh)
ファイルに接続したノード間で同期をとる．

2.3.1 集団的グローバル I/O

集団的グローバル I/O は集団実行による順番探査でファイルを操作する．入出力項目が分散配列のとき，ファイルポインタは読み込まれた，または，書き出された分散配列に対応付けられたテンプレートのサイズだけ先に移動する．入出力項目が重複変数のとき，ファイルポインタは読み込まれた，または，書き出された重複変数のサイズだけ先に移動する．

【Fortran】

入力項目が分散配列の全体配列のとき，分散配列のテンプレートに対応して設定されたファイルビューに沿ってファイルから読み出された値が各ノードのもつ配列要素に設定される．入力項目が重複変数のとき，ファイルから読み出された値がすべての実行ノードの変数に設定される．

出力項目が分散配列の全体配列のとき，各ノードのもつ配列要素の値が分散配列のテンプレートに対応して設定されたファイルビューに沿ってファイルに書き出される．出力項目が重複変数，または，一般の式のとき，実行ノードの 1 ノードで評価された値がファイルに書き出される．

【C】

ベース言語が C のとき，以下のライブラリ関数を用いて集団的グローバル I/O する．

xmp_file_set_view(xmp_file_t *fh, long long disp, xmp_array_t ap, xmp_range_t *rp)
ファイルビューを設定する．

xmp_file_clear_view_all(xmp_file_t *fh, long long disp)
ファイルビューを初期状態に戻す．

xmp_fread_all(xmp_file_t *fh, void *buff, size_t size, size_t count)
全実行ノードが同時に共有ポインタの位置からデータを読み込む．

xmp_fwrite_all(xmp_file_t *fh, void *buff, size_t size, size_t count)
全実行ノードが同時に共有ファイルポインタの位置へデータを書き出す．

xmp_fread_darray_all(xmp_file_t *fh, xmp_array_t ap, xmp_range_t *rp)
共有ファイルポインタに位置から分散配列へデータを読み込む．

xmp_fwrite_darray_all(xmp_file_t *fh, xmp_array_t ap, xmp_range_t *rp)
分散配列のデータを共有ファイルポインタの位置へ書き出す．

2.3.2 排他的グローバル I/O

排他的グローバル I/O は各ノードが排他的に順番探査でファイルを操作する．排他的グローバル I/O における OPEN 文，CLOSE 文は集団実行であるが，READ 文，WRITE 文は各ノードが独立に実行する．

排他的グローバル I/O で接続されたファイルは，その接続を実行したノード集合で排他的グローバル I/O で接続を解除することができる．排他的グローバル I/O における入出力文は各ノードから早くファイルにアクセスできたノードの順に排他的に読み込む，または，書き出す．

【Fotran】

入出力文の単位は 1 つの READ 文，または，WRITE 文である．排他的グローバル I/O における OPEN 文の POSITION 指定子で接続時のファイルポインタの位置が決まる．接続後は各ノードによって入出力したデータサイズだけ先に移動する．

【C】

ベース言語が C のとき，以下のライブラリ関数を用いて排他的グローバル I/O する．

xmp_fread_shared(xmp_file_t *fh, void *buff, size_t size, size_t count)
共有ファイルポインタの位置からデータを読み込む．

xmp_fwrite_shared(xmp_file_t *fh, void *buff, size_t size, size_t count)
共有ファイルポインタの位置へデータを書き出す．

Vol.2011-ARC-197 No.36
Vol.2011-HPC-132 No.36
2011/11/29

2.3.3 位置指定グローバル I/O

位置指定グローバル I/O は各ノードが独立に直接探査でファイルを操作する．実行ノードで共有するファイルポインタを各ノードでそれぞれ独立に移動させて入出力する．

【Fotran】

位置指定グローバル I/O における OPEN 文，CLOSE 文は集団実行であるが，READ 文 WRITE 文は各ノードが独立に実行する．

位置指定グローバル I/O における OPEN 文で接続されたファイルは，その OPEN 文を実行したノード集合で排他的グローバル I/O における CLOSE 文だけで接続を解除することができる．

位置指定グローバル I/O における入出力文は REC 指定子で指定したファイルポインタの位置が決まる．ファイルポインタの位置は位置指定グローバル I/O における OPEN 文の RECL 指定子の値と入出力文の REC 指定子の値をかけ合わせた値だけファイルの先頭から先に移動する．

【C】ベース言語が C のとき，以下のライブラリ関数を用いて位置指定グローバル I/O する．

xmp_fread(xmp_file_t *fh, void *buff, size_t size, size_t count)

固有ファイルポインタの位置からデータを読み込む．

xmp_fwrite(xmp_file_t *fh, void *buff, size_t size, size_t count)

固有ファイルポインタの位置へデータを書き出す．

3. XcalableMP と NetCDF ファイルのインタフェース

本章では科学アプリケーションでよく利用されている NetCDF(Network Common Data Form)[8] ファイルについて説明する．また，XcalableMP の拡張機能として設計中である XcalableMP から NetCDF ファイルを操作するライブラリインタフェースについて説明する．

3.1 NetCDF

NetCDF は API とファイルフォーマットで構成されるデータを格納するのためのパッケージである．NetCDF はプラットフォームに依存しないフォーマットであるため可搬性が高い．NetCDF API はその可搬性の高い NetCDF ファイルを操作するインタフェースである．NetCDF API はシングルプロセスでは使い勝手のよいインタフェースであるが，並列に操作するインタフェースがない．

3.2 インタフェース

Parallel NetCDF(PnetCDF) は NetCDF ファイルを並列に操作可能であるため，並列言語で使えると，入出力やデータ転送に費やす時間が少なく効率的である．しかし，PnetCDF は MPI のデータ分散のプログラムモデルに基づいて設計されているため，XcalableMP で PnetCDF を直接呼ぶことはできない．

我々は XcalableMP のプログラムモデルにあわせて NetCDF ファイルを操作する XcalableMP の拡張機能を設計中である．この拡張機能は XcalableMP から PnetCDF API を呼び NetCDF ファイルを操作するライブラリインタフェースである．

PnetCDF API によるデータ書き込みの例を図 6 に，PnetCDF API を呼ぶ XMP-IO ライブラリによるデータ書き込みの例を図 7 に示す．図 6，図 7 はグローバルビューの配列として GA(6,4)（ローカルの配列としては A(3,2) を $(2*2)$ の 2 次元に配置したプロセッサに各次元 block 分散させたものをファイルに出力するプログラムである．

図 6，図 7 に示すように XcalableMP プログラムではファイルに配列の値を書き出す関数 xmpnc_vara_int_all() に各ノードが書き出すグローバルビューでの配列の位置や長さを指定しなくてよい．XcalableMP のプログラム例ではグローバルビューで配列を宣言している．グローバルビューでの配列の位置や長さの情報は xmp_array_of() 関数で取り出せる配列情報に含まれる．このように PnetCDF を使うよりも一般的に短く簡単になる特徴がある．

図 6，図 7 の例は配列全体の書き出しであったため出力する範囲を指定してないが，部分配列を書き出すときのそれぞれの関数を図 8 に示す．部分配列を入出力するとき PnetCDF は配列全体を入出力する図 6 と同様である．部分配列を入出力する XcalableMP 拡張関数には入出力する配列の幅を rp に指定しなければならない．このように XcalableMP の拡張ライブラリで PnetCDF の API を呼ぶインタフェースの設計は PnetCDF のすべての API に対して関数名だけを変更するか，引数を変更するか，関数を増やすか，サポートする必要がないかを決めなければならない．今後，すべての PnetCDF API に対する XcalableMP のインタフェースを考え，拡張機能（ライブラリインタフェース）を実装する予定である．

4. 並列入出力の評価

並列入出力可能なファイルシステム Lustre 上で以下のプログラムの入出力性能を評価した．ベース言語はすべて C である．括弧内の文字は以降の図の凡例や結果の説明に用いる．

- MPI-IO(MPI-IO)
- Parallel netCDF(PnetCDF)

Vol.2011-ARC-197 No.36
Vol.2011-HPC-132 No.36
2011/11/29

```
                ⋮
#define X 3
#define Y 2
#define XPROC 2
#define YPROC 2
                ⋮
        int A[X][Y];
        int dimids[2], varid;
        MPI_Offset start[2], count[2];
                ⋮
        ncmpi_create(mpi_comm, filename, NC_CLOBBER, MPI_INFO_NULL, &ncid);
        ncmpi_def_dim(ncid, "x", XPROC*X, &(dimids[0]));
        ncmpi_def_dim(ncid, "y", YPROC*Y, &(dimids[1]));
        ncmpi_def_var(ncid, "data", NC_INT, 2, dimids, &varid);
        ncmpi_enddef(ncid);
        for (di = 0; di ¡ XPROC*YPROC; di++) {
            start[0] = rank / YPROC * X;
            start[1] = (rank % YPROC) * Y;
        }
        count[0] = X;
        count[1] = Y;
        ncmpi_put_vara_int_all(ncid, varid, start, count, &A[0][0]);
        ncmpi_close(ncid);
                ⋮
```

図 6 PnetCDF の書き出しプログラム例

- データ入出力を C，データ転送を MPI のワンサイド通信 (MPI)
- XcalableMP IO(XMP-IO)

4.1 実験環境

計算機は各ノード INTEL Xeon X5670 2.93GHz (2 ソケット 12CPU コア) で，88 ノード構成の並列クラスタを使用した．使用したファイルシステムは Lustre であり，ストライプ数は実験的に安定した性能がでる 24 とした．ストライピングサイズはデフォルト値の 1[MB] とした．すべての実行において 1 ノードあたり 1 プロセスで実行した．入出力の実

```
                ⋮
#define X 6
#define Y 4
#define XPROC 2
#define YPROC 2
!#pragma xmp nodes p(XPROC,YPROC)
!#pragma xmp tmplate t(0:X-1, 0:Y-1))
!#pragma xmp distribute t(block, block) onto p
int GA[M][N];
!#pragma xmp align GA[i][j] with t(i,j)
                ⋮
        int dimids[2], varid;
        xmp_array_t ap;
        ap = xmp_array_of(GA);
        xmpnc_create(mpi_comm, filename, NC_CLOBBER, MPI_INFO_NULL, &ncid);
        xmpnc_def_dim(ncid, "x", X, &(dimids[0]));
        xmpnc_def_dim(ncid, "y", Y, &(dimids[1]));
        xmpnc_def_var(ncid, "data", NC_INT, 2, dimids, &varid);
        xmpnc_enddef(ncid);
        xmpnc_put_vara_int_all(ncid, varid, ap);
        xmpnc_close(ncid);
                ⋮
```

図 7 XMP-IO のライブラリを用いた書き出しプログラム例

行時間は同時にクラスタ利用した他のユーザの使用方法に大きく影響を受ける．すべての実行結果において，3 度実行した最も性能の高い値を結果とした．

C コンパイラは Intel C コンパイラ，MPI は mvapich1.7 を用いた．書き出しにおける MPI は get でデータを集めて 1 ノードから書き出した，読み込みにおける MPI は 1 ノードでファイルを読み込んでから put でデータを各ノードに転送した．XcalableMP は筑波大学が開発している Omni XcalableMP コンパイラ（開発バージョン）を使用した．XcalableMP プログラムは並列に入出力可能な集団的グローバル I/O の xmp_fwrite_darray_all() 関数と xmp_fread_darray_all() 関数で記述した．Lustre のストライブサイズは安定して高い性能

Vol.2011-ARC-197 No.36
Vol.2011-HPC-132 No.36
2011/11/29

PnetCDF

```
ncmpi_put_vara_int_all(ncid, varid, start[], count[], *op);
        int ncid, varid
        const MPI_Offset start[], count[]
        const int *op
```

XMP-IO

```
xmpnc_put_varas_int_all(ncid, varid, ap, rp);
        int ncid, varid
        xmp_array_t ap
        xmp_range_t rp
```

図 8 部分配列を出力する PnetCDF と XMP-IO の API

が出ている 24 とした．

4.2 書き出し性能

各ノードに整数型の $10,000 * 1,000$ の 2 次元配列を置き，それぞれの配列が $(proc * proc)$ のブロック分散されて配置されていたように，ファイルへ書き出す（$proc = 2, 3, 4, 5$）．例として $proc = 2$ で各ノードに格納する配列が A[2][3] のときのデータ分散とファイルの関係を図 9 に示す．

すべてのノードが保持する配列をファイルへ書き出したときの書き出し性能と計算ノード数の関係を図 10 に示す．ここで，書き出し性能とはファイルに書き込んだデータ量に対するデータ転送と出力文に費やした時間である．

MPI を除く MPI-IO，PnetCDF，XMP-IO はノード数が増えたとき，スケールしている．MPI は 1 ノードに集めてディスクに書き出すため 1 ノードからの Lustre ファイルシステムへ書き出す性能がボトルネックになっている．一方，並列に書き出し可能な MPI-IO，PnetCDF，XMP-IO の書き出し性能はどれも 25 並列までスケールし，近い性能となっている．PnetCDF が MPI-IO や XMP-IO よりも 1 割程度遅くなっているのは，派生データ型をつくるのに時間を費やしたか，派生型を作らずにつくらずに書き出したため MPI-IO で最適化されなかったと考えられる．25 並列で XMP-IO が MPI-IO の性能を上回っているが，これは他のユーザの計算機の利用による外乱と考えられる．今回のプログラムにおいて，XMP-IO が MPI-IO を上回る書き方はされていない．

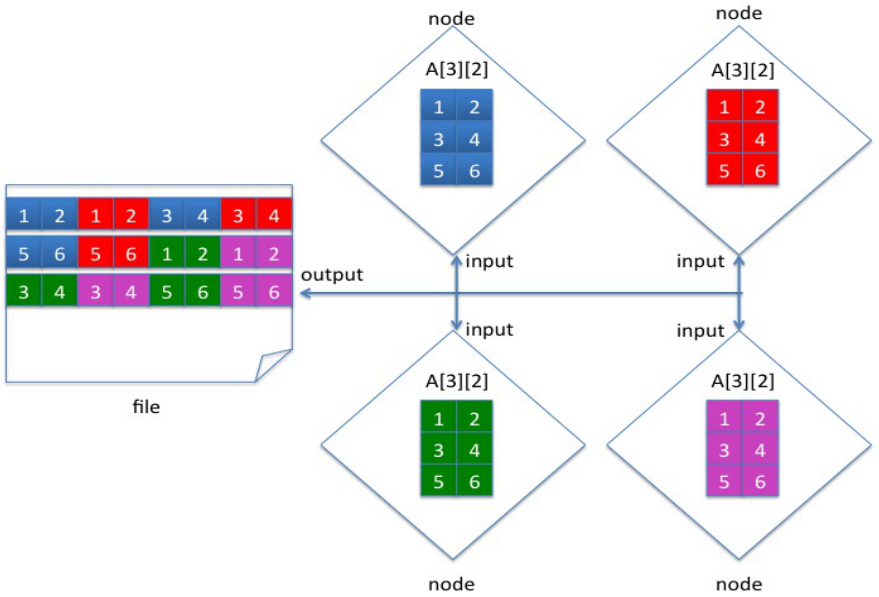

図 9 ファイルへの書き出しとファイルからの読み込み例

4.3 読み込み性能

整数型のデータが (ノード数 *10,000,000) 個書かれたファイルを $(proc * proc)$ の 2 次元ブロック分散した配列に読み込んだ．ファイルを配列に読み込んだときの読み込み性能と実行ノード数の関係を図 11 に示す．ここで，読み込み性能とはファイルから読み込んだデータ量に対するファイルから読み込み各ノードの配列に値を設定するまでの時間である．

MPI-IO，PnetCDF，XMO-IO は同等の性能で読み込まれた．

5. おわりに

本稿では XcalableMP の仕様の一部として定義されている並列 IO インタフェース XMP-IO について説明した．我々はよく使われる NetCDF ファイルを XcalableMP から並列に入出力するためのライブラリインタフェースを設計した．これらの XcalableMP の並列入出力機能について，並列ファイルシステム Lustre 上で並列入出力を評価し，その有効性を確か

Vol.2011-ARC-197 No.36
Vol.2011-HPC-132 No.36
2011/11/29

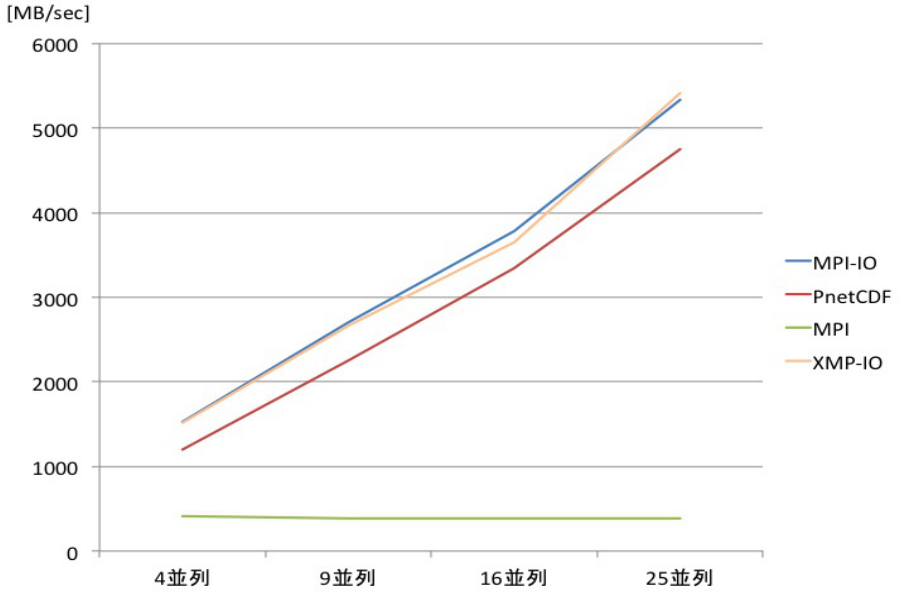

図 10 書き出し性能と実行ノード数の関係

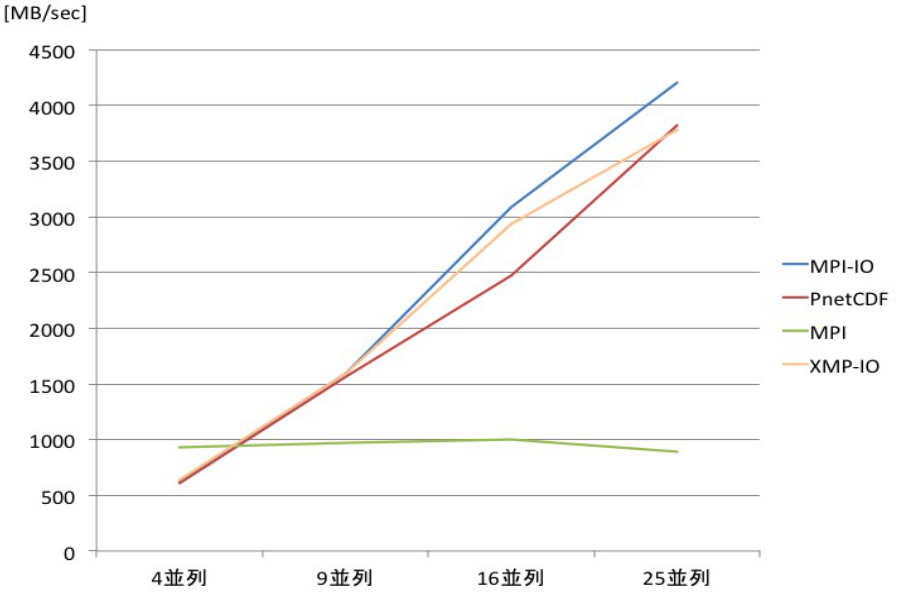

図 11 読み込み性能と実行ノード数の関係

めた．

今後，データ量を増やした実験と cyclic 分散や block-cyclic 分散の配列への読み込み，および，書き込み性能を評価する．また，XcalableMP から PnetCDF API を呼び NetCDF ファイルを操作するライブラリインタフェースを検討し実装する予定である．

謝辞 本研究の一部は，文部科学省「e-サイエンス実現のためのシステム統合・連携ソフトウェアの研究開発」における課題「シームレス高生産・高性能プログラミング環境」による．また，XcalableMP の仕様は，次世代並列プログラミング言語検討会による．

参 考 文 献

1) High Performance Fortran Forum,: High Performance Fortran Language Specification Version 2.0, http://hpff.rice.edu/versions/hpf1/hpf-v11/hpf-v11.pdf (1997).
2) Reid, J.,: Coarrays in the next Fortran Standard, *ISO/IEC JTC1/SC22/WG5 N1787* (2009).
3) UPC Consortium, UPC Specifications v1.2, *Technical report, Lawrence Berkeley National Lab (LBNL-59208)* (2005).
4) Callahan D., Chamberlain B., and Zima H., The Cascade High Productivity Language, *In Proc. 9th Int'l. Workshop on High-Level Parallel Programming Models an Supportive Environments (HIPS 2004)*, pp. 52—60 (2004).
5) Yang, M., McGrath, E. R., Folk, M.,: Atomospheric sciences and climate applications using HDF and HDF5, *Conference on Interactive Information Processing Systems* (2005).
6) Yan, M., Mcgrath, E. R., Folk, M.,:Performance study of HDF5-WRF IO modules, , *WRF Workshop* (2004).
7) WRF: http://wrf-model.org/
8) Rew, R. K., Davis, G. P.: The unidata netCDF: Software for scientific data access. *In: Sixth International Conference on Interactive Information and Processing Systems for Meteorology, Oceanography, and Hydrology, American Meteorology Society*, pp. 33-40 (1990).
9) Li, J., Liao, W. K., Choudhary, A., Ross, R., Thakur, R., Gropp, W., Latham, R., Siegel, A., Gallagher, B., Zingale, M.: Parallel netCDF: A high-performance scien-

Vol.2011-ARC-197 No.36
Vol.2011-HPC-132 No.36
2011/11/29

tific I/O interface. *In: SC ' 03: Proceedings of the 2003 ACM/IEEE Conference on Supercomputing, IEEE Computer Society*, p(2003)
10) A parallel API for creating and reading NetCDF file, (2010).
11) 村井 均: 分散メモリ向け並列プログラミング言語における効率的な並列化に関する研究, 博士論文, 筑波大学システム情報工学研究科 (2010).