社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－HPC－108 (6)
2006／10／6

Parallel netCDF インタフェースによる計算機間集団型入出力機能の実装

辻 田 祐 一†

netCDF (Network Common Data Form) は、計算アプリケーションにおける多次元データに対し，自己記述的で可搬性の高いデータ形式をサポートする入出力インタフェースを定めており，C 及び Fortran インタフェースが提供されている．さらに近年，並列入出力機能をサポートした parallel netCDF が開発されている．parallel netCDF では，netCDF のデータ形式をサポートしつつ，MPI-I/O ライブラリを用いる事で並列入出力の機能を実現している．しかし，異なる MPI ライブラリを持つ計算機間ではこの機能は利用できない．このような環境でもこの機能を実現するために，種類の異なる計算機間で透過的な MPI-I/O 操作を実現する Stampi を用い，数種類の関数について機能の実装を行った．さらに実装した機能の性能をネットワークで繋がれた PC クラスタ間で計測を行い，データ長が大きい時には十分利用可能である事を確認した．

Implementation of a Parallel NetCDF Interface for Collective Remote I/O

YUICHI TSUJITA†

In scientific applications, netCDF was developed to support a view of data as a collection of self-describing, portable, and array-oriented objects that can be accessed through a simple interface. It also supports C and Fortran interfaces to access a netCDF file. Recently parallel netCDF has been developed with the help of an MPI-I/O library to realize parallel I/O with netCDF data format support. Unfortunately, the same operations in remote I/O between different MPI libraries have not been realized. Stampi realizes seamless MPI communications and MPI-I/O operations both inside a computer and among computers. Its remote I/O mechanism has been introduced in several parallel netCDF interfaces to realize such operations. The newly implemented mechanism has been evaluated on two interconnected PC clusters, and sufficient performance has been achieved with huge amount of data.

1. はじめに

近年の計算機の処理速度の向上と計算機資源の大規模化により，大規模な並列計算が行われるようになった．これに伴い，入出力されるデータも大規模化している．そのような大規模データに対し効率的に入出力を行うために，並列入出力機能が必要になってきた．様々な並列入出力機能の実装がある中で，並列処理における標準的な通信インタフェースである MPI[1] では，MPI-I/O と呼ばれる並列入出力インタフェースが定められている[2]．

MPI を実装した様々なライブラリにおける入出力は，同じライブラリ内でのみ利用可能であり，異なる MPI ライブラリを持つ別の計算機への入出力は出来ない．並列計算で出力されるデータは，可視化などの目的で，別の計算機に転送される事があるが，計算機間のデータ転送は，例えばプログラムの実行後にデータ転送を行うなど，計算プログラムとは別の処理で行ってきた．このような計算機間入出力機能を計算プログラム内で実現する為，Stampi-I/O[3] が Stampi[4] の拡張機能として実現された．Stampi は日本原子力研究開発機構 システム計算科学センターで開発された異機種計算機間 MPI ライブラリで，当初は異なる MPI ライブラリを持つ計算機間の透過的な MPI 通信を実現するために開発された．Stampi-I/O は Stampi の通信基盤を利用した計算機間 MPI-I/O 機能を提供しており，リモート計算機への入出力は，入出力操作を行う MPI-I/O プロセスをターゲットの計算機上に起動することにより実現している．利用者は MPI-I/O のインタフェースを通して，自由に異なる計算機への入出力操作が可能である．MPI-I/O プロセスは通常，起動された計算機が持つ計算機ベンダ提供の MPI-I/O ライブラリを用いて入出力を行うが，これが利用できない場合，UNIX I/O を用いた入出力を行う．

数値計算プログラムでは，データ形式が固定のものより，必要に応じてデータ形式の変更が可能な可搬性のあるデータ形式が有用である．そこで，プラットホームに依存しない可搬性のある入出力データの形式

† 近畿大学 工学部 電子情報工学科
Department of Electronic Engineering and Computer Science, Faculty of Engineering, Kinki University

や入出力インタフェースを標準化し，それらを用いた透過的な入出力操作を実現する取り組みがいくつか行われているが，その一つとして netCDF[5] と呼ばれるインタフェース仕様がある．netCDF は多次元データを自己記述的でかつ可搬性の高いデータ形式として定義し，新規データの作成や保存データへのアクセスにおいて，可搬性のある入出力を提供することを目的に策定されたインタフェース仕様である．このようなデータ形式を共有することにより，データ管理コストの低減とデータ相互参照の効率向上が期待できる．C や Fortran などのインタフェースが提供されており，netCDF のインタフェースを通して可搬性のある入出力操作が可能である．さらに近年，並列入出力へ拡張した parallel netCDF[6]（以下，PnetCDF と記す.）が開発され，並列入出力機能も利用可能である．PnetCDF では，並列入出力の実装に MPI-I/O を利用しており，例えば，代表的な MPI-I/O ライブラリである ROMIO[7] との組合せで利用可能である．しかし，異なる MPI ライブラリを持つ計算機間での入出力は出来ない．

そこで，Stampi の計算機間 MPI-I/O 機能である Stampi-I/O を用い PnetCDF のインタフェースを通したリモート入出力操作の実装を行った．以下，本実装の詳細と，基本機能の性能評価の結果について述べた後，今後の課題について述べる．

2. PnetCDF インタフェースを用いたリモート入出力

本章では，Stampi-I/O を用いて実装した PnetCDF インタフェースによるリモート入出力機能のアーキテクチャと動作メカニズムについて説明する．

2.1 PnetCDF インタフェースへの対応

PnetCDF には，様々な入出力インタフェースが用意されており，それらのインタフェース関数の中では，数種類の MPI 関数が利用されている．いくつかの PnetCDF インタフェースに関してそれらの内部で呼び出される MPI 関数との対応を**表 1** に示す．オリジナルの PnetCDF では，MPI 関数の部分を計算機ベンダが提供する MPI ライブラリや ROMIO などを利用している．このような場合，自計算機内で利用することは出来るが，異なる MPI ライブラリを持つ別の計算機への入出力は利用出来ない．そこで，Stampi が提供する MPI インタフェースを用いて機能拡張を行った．表 1 に示すような PnetCDF で用いている MPI 関数は Stampi でサポートしており，PnetCDF のライブラリを作成する際に Stampi のライブラリをリンクするように変更し，実装を行った．

表 1 PnetCDF インタフェースから呼び出される MPI 関数の一例

PnetCDF 関数	呼び出される MPI 関数
ncmpi_create()	MPI_File_open(), MPI_File_delete() など
ncmpi_open()	MPI_File_open(), MPI_File_delete() など
ncmpi_put_var_int()	MPI_Comm_rank(), MPI_File_set_view(), MPI_Type_hvector(), MPI_Type_commit(), MPI_Type_free(), MPI_File_write() など
ncmpi_close()	MPI_Allreduce(), MPI_File_close() など

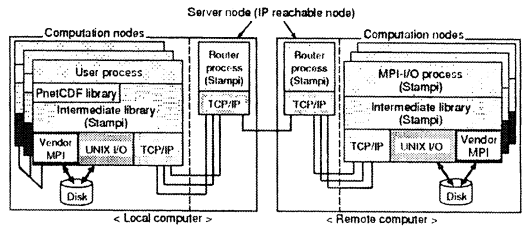


図 1 PnetCDF インタフェースをサポートする計算機間入出力機能のアーキテクチャ

2.2 入出力機能のアーキテクチャ

入出力機能のアーキテクチャを**図 1** に示す．計算機内部のユーザ・プロセス間はベンダ提供の MPI ライブラリを用いた高速な通信が行われる．PnetCDF インタフェースを用い自計算機上で入出力を行う場合，PnetCDF 関数の内部で関連する Stampi の MPI-I/O インタフェースが呼び出され，このインタフェースを通して，計算機ベンダ提供の MPI-I/O ライブラリが利用可能な場合，これを用いた高速な入出力が，利用できない場合は，UNIX I/O を用いた入出力操作が行われる．

一方，異なる計算機との通信では，通信相手の計算機にユーザ・プロセスを起動し，TCP ソケットにより接続された通信経路を用いて MPI 通信を行う．また，各計算ノードが直接外部と通信が出来ない場合は，管理ノードのような外部と直接通信可能なノードに通信中継プロセス（router process）を起動し，これを経由して通信を行う．計算機間入出力の際に起動される MPI-I/O プロセスは，ファイルのオープン時に起動される．MPI-I/O プロセスによる入出力では，自計算機内部での入出力操作と同様，Stampi の MPI-I/O インタフェースを通して，起動した計算機のベンダ提供の MPI-I/O ライブラリを用いた入出力が行われる．この MPI-I/O ライブラリが利用できなければ，Stampi が独自に用意している UNIX I/O 関数による入出力ライブラリを利用する．

2.3 動作メカニズム

PnetCDF インタフェースを用い入出力を行うプログラムの起動メカニズムについて**図 2** を用いて説

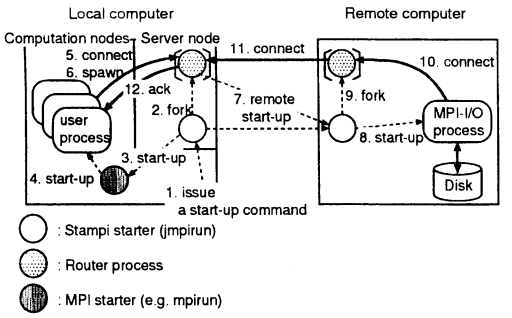


図 2 PnetCDF インタフェースを用いた計算機間入出力の動作メカニズム

明する．まず始めに Stampi の起動コマンドにより，Stampi の起動プロセス（Stampi starter）が起動される．次に Stampi starter 自身は，ネイティブな MPI ライブラリの起動プログラム（MPI starter; 例えば mpirun）を起動し，この起動プログラムによりユーザ・プロセスが起動される．ユーザ・プロセス内部で，`ncmpi_create()`（ファイル新規作成時）又は `ncmpi_open()` が呼び出されると，まず始めにこれらの関数内部にある `MPI_File_open()` が呼び出される．この関数呼び出しにより，対応する Stampi の MPI インタフェースが呼び出され，`info` オブジェクトに指定した計算機上に MPI-I/O プロセスを起動する．なお，プロセス起動においては，リモートシェルコマンド（rsh 又は ssh）を用いている．ユーザ・プロセスがあるノードがこのプロセスを起動するリモート計算機のノードと直接通信出来ない時は，外部と通信可能なノードに通信中継プロセス（router process）を起動し，これを介してプロセス起動を行う．MPI-I/O プロセスが起動されたリモート計算機側も同様に，MPI-I/O プロセスがユーザ・プロセスと直接通信出来ない場合，外部と通信可能なノードに通信中継プロセスを起動する．この処理の後に，ユーザ・プロセスと MPI-I/O プロセスの間で通信経路が形成され，これを通してユーザ・プロセスから入出力命令が MPI-I/O プロセスに送られることにより入出力操作が行われる．

入出力操作の後，`ncmpi_close()` が呼び出されると，内部で `MPI_File_close()` が呼び出され，この入出力要求が MPI-I/O プロセスに送られる．要求を受けた MPI-I/O プロセスはファイルを閉じ，自分自身が消滅することにより，入出力操作が完了する．

3. 性能評価

本実装について，ネットワークで繋がれた PC クラスタ間で入出力性能を評価した．それぞれの PC クラスタの仕様を**表 2** に示す．それぞれの PC クラスタには 1 台の管理ノードと 4 台の計算ノードがあり，ノード間はギガビットのイーサネットスイッチを介し，1 Gbps の帯域で繋いだ．PC クラスタ II には，PVFS2[8]（version 1.3.2）を導入し，4 台の計算ノードからそれぞれ 73 GByte のディスク領域を束ねて管理ノード上に 292 GByte の並列ファイルシステムを構築した．また PC クラスタ間はそれぞれのイーサネットスイッチを Allied Telesis CentreCOM GS908GT を介して 1 Gbps の帯域で繋いだ．

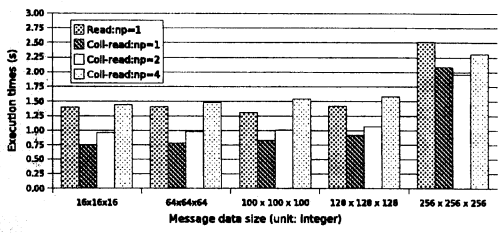
本試験では，PC クラスタ I から PC クラスタ II の PVFS2 ファイルシステムに対し，PnetCDF のインタフェースから Stampi の MPI-I/O インタフェースを呼び出した場合の計算機間入出力の評価を行った．

評価試験では，基本データ型を integer とし，16×16×16, 64×64×64, 100×100×100, 128×128×128, 256×256×256 の 5 種類（それぞれ，約 16 KByte, 1 MByte, 3.8 MByte, 8 MByte, 64 MByte に相当）の 3 次元データを用い，測定を行った．集団型入出力操作においては，3 次元データの x，y，z 軸の内，z 軸において，プロセス間で等しいサイズにデータをブロック分割し，入出力操作を行った．リモート計算機への入出力に必要な情報は `MPI_Info_set()` により `info` オブジェクトに設定された．書き込み操作の場合，これを用い `ncmpi_create()` により netCDF データを新規作成し，さらにデータに関連する情報を `ncmpi_put_att_text()`, `ncmpi_def_dim()`, `ncmpi_def_var()` などにより登録した後に，`ncmpi_enddef()` により，設定モードを終了する．次に `ncmpi_put_vars_int_all()` により integer（`NC_INT` を使用）のデータ型で書き込んだ．一方，読み出しの場合，`ncmpi_open()` によりファイルをオープンし，さらに当該ファイルに記録されているデータの情報を `ncmpi_inq_varid()` や `ncmpi_inq_dimid()` などで取り出し，次に `ncmpi_get_vars_int_all()` によりデータを読み出した．いずれの場合もファイルは，`ncmpi_close()` により閉じられ，入出力処理を終了した．

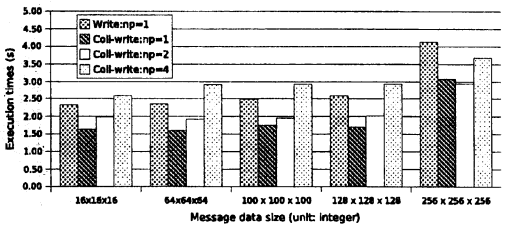
時間計測は，ファイルをオープンしてから入出力を行い，ファイルを閉じるまでの操作を 20 回繰り返し，ファイルのオープンから閉じるまで（`ncmpi_create()` 又は `ncmpi_open()` から `ncmpi_close()` まで）と入出力のみ（例えば `ncmpi_put_vars_int_all()`）の最小時間をデータとして記録した．測定結果を**図 3** に示す．この図で，"`Read`", "`Write`" とあるのが，比較対象として実施した非集団型の読み出し（`ncmpi_get_var_int()`）並びに書き込み操作（`ncmpi_put_var_int()`），"`Coll-read`", "`Coll-write`" とあるのが，上記の集団型の読み出し並びに書き込み操作を行った場合のファイルをオープンしてから閉じるまでに要した入出力全体に要した時間である．また，横にある数字はユーザ・プロセスの数であり，MPI-I/O プロセスも同じ数だけ起動した．データ長

表 2 使用した二つの PC クラスタの仕様

	PC クラスタ I	PC クラスタ II
管理ノード	Dell PowerEdge800 1 台	Dell PowerEdge1600SC 1 台
計算ノード	Dell PowerEdge800 4 台	Dell PowerEdge1600SC 4 台
CPU	Intel Pentium-4 3.6 GHz × 1	Intel Xeon 2.4 GHz × 2
チップセット	Intel E7221	ServerWorks GC-SL
メモリ	1 GByte DDR2 533 SDRAM	2 GByte DDR 266 SDRAM
ディスク装置	80 GByte (Serial ATA) × 1 (管理ノード) 80 GByte (Serial ATA) × 1 (計算ノード)	73 GByte (Ultra320 SCSI) × 1 (管理ノード) 73 GByte (Ultra320 SCSI) × 2 (計算ノード)
NIC	Broadcom BCM5721 (PCI-Express : オンボード)	Intel PRO/1000-XT (PCI-X ボード)
スイッチ	3Com SuperStack3 Switch 3812	3Com SuperStack3 Swicth 4900
OS	Fedora Core 3	
カーネル	2.6.12-1.1381_FC3smp (管理ノード) 2.6.11 (計算ノード)	2.6.12-1.1381_FC3smp (管理ノード) 2.6.11-1SCOREsmp (計算ノード)
ネットワークドライバ	Broadcom tg3 version 3.58b	Intel e1000 version 6.0.54 (管理ノード) Intel e1000 version 5.6.10.1 (計算ノード)
MPI ライブラリ	MPICH version 1.2.7p1	

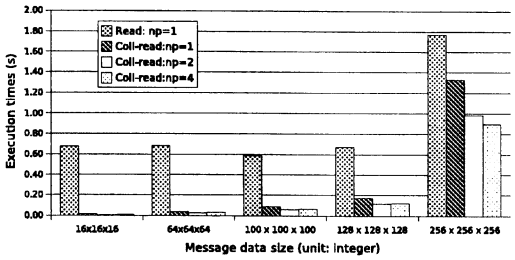


(a) 読み出し

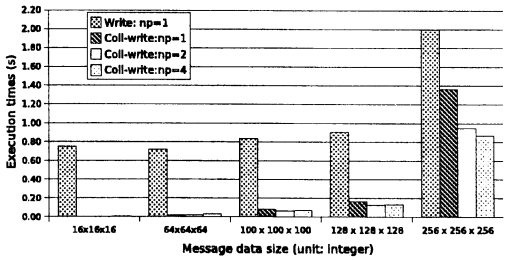


(b) 書き込み

図 3 integer 型を用いた PnetCDF インタフェースによる PC クラスタ間の入出力に要した時間



(a) 読み出し



(b) 書き込み

図 4 integer 型を用いた PnetCDF インタフェースによる PC クラスタ間のデータ入出力のみに要した時間

が 128×128×128 までは，非集団型の入出力の処理時間に比べ，集団型入出力で 1 並びに 2 プロセスの場合の処理時間の方が短いことが分かった．またデータ長が 256×256×256 の場合，4 プロセスで集団型入出力を行う場合でも，非集団型入出力を行うよりも処理時間が短くなることを確認した．以上の結果より，データ長が長ければ，集団型入出力が有効であることが分かった．

次に，データ入出力のみに要した時間を図 4 に示す．読み出し，書き込み操作の両方において，集団型入出力においては，データ長が短い場合，図 3 に示した入出力全体に要する時間に比べて，非常に短い時間で入出力が終了していることが分かる．一方，非集団型ではデータ長が短くてもある一定の処理時間がかかっていた．これは，集団型では PVFS2 ファイルシステムに対し，循環バッファ機構を用いているのに対し，非集団型では，単一バッファによる入出力を行っているためと考えられる．また，両操作において，集団型入出力においては，プロセス数の増大に伴い，処理時間が短縮されていることも分かった．

図 3 と図 4 の結果を比較すると，データ入出力以外に要する時間が割と大きいことが分かった．PnetCDF で入出力を行う場合，実データの入出力を行う前にアクセスする実データに関する情報をファイルから読み出したり，書き込んだりするなど，実データの入出力

以外に必要な処理があり，この増加した時間はこれらの処理によるものと考えられる．これらの処理の中では，データ長の短いデータを MPI_Allreduce() のような割と通信コストの高い集団通信関数を用いデータ通信を数多く行っていたり，プロセス間の同期を取る為に頻繁に MPI_Barrier() を多用していることが，処理時間が無視できなくなるほど大きくなっている原因と考えられる．特に，入出力全体の処理に要する時間は，プロセス数が 2 プロセスまでは短縮され，4 プロセスで逆に増えるという結果であったため，入出力のみに要した時間を考慮すると，入出力操作以外の処理で，2 プロセスから 4 プロセスに増える際に集団型データ通信などにより，処理時間が増えたものと考えられる．

4. 関連研究

MPI-I/O ライブラリの実装の代表的なものである ROMIO[7] は様々なファイルシステムへの MPI-I/O 機能を ADIO[9] を通して実現している．ADIO は様々なファイルシステムへのインタフェースを用意し，上位のインタフェースに対して一様なインタフェースを提供することで個々のファイルシステムのアーキテクチャの違いを意識させない実装になっている．さらに MPICH[10] が利用できるリモート計算機への入出力操作をサポートする機能も実装されている[11]．しかし異なる MPI ライブラリ間での入出力はサポートされていない．

異なる MPI ライブラリ間での MPI 通信並びに MPI-I/O 機能をサポートしたものとして，Stampi 以外に PACX-MPI[12] や GridMPI[13] が挙げられる．PACX-MPI は，それぞれの計算機の MPI ライブラリの上位にインタフェースライブラリを配置し，それぞれの計算機に通信を中継するプロセスを起動して，クラスタ内，クラスタ間の MPI 通信を実現している．グリッド環境においての使用を目的に開発された GridMPI は，クラスタ内は YAMPII[14] で実装されている 1 対 1 (P2P) 通信方式あるいはベンダ MPI ライブラリによる P2P 通信で，クラスタ間は IMPI[15] に準拠した P2P 通信で MPI 通信を実現している．一方，Stampi では，それぞれの計算機の MPI ライブラリの上位にインタフェースライブラリを配置する実装は PACX-MPI と同様であるが，計算ノードが外部と直接通信可能な計算機には通信中継プロセスを起動しない柔軟な実装を行っている．

数値計算プログラムで発生する大規模データを異なるプラットフォームでも透過的に利用できるために，ポータブルなデータ形式を定めた入出力インタフェースが定められている．その代表的なものとして，netCDF[5] と HDF5[16] がある．netCDF は，多次元データを異なるアーキテクチャを持つ計算機間でも透過的に効率良く扱うための可搬性の高いデータ形式を定めており，netCDF のインタフェースを通して自由にアクセスすることができる．一方，HDF5 は，階層構造を持つデータ形式を定めており，大規模データを効率良く扱うために開発され利用されている．HDF5 では，Dataset と Group という 2 つのオブジェクトがあり，Dataset では，データ要素の多次元配列を扱い，Group では HDF5 ファイルのオブジェクトの組織化のための仕組みを提供している．

netCDF において並列入出力に対応させたものとして，parallel netCDF[6] が開発されている．parallel netCDF インタフェースの下層レイヤには MPI-I/O インタフェースが利用されており，計算機ベンダが提供する MPI-I/O ライブラリあるいは ROMIO を用いる事で並列入出力において netCDF 形式のデータを扱うことが可能になっている．HDF5 においても MPI-I/O インタフェースを利用した並列入出力インタフェースが開発されており，PVFS[17] が利用可能な Linux が稼働する PC クラスタで，HDF5 から MPI-I/O，PVFS I/O の順に階層的にライブラリを呼び出す形で並列入出力機能を実現している[18]．

5. まとめと今後の課題

Stampi が持つ計算機間入出力機能を用い，PnetCDF インタフェースによる計算機間入出力機能の実装の試みを行い，計算機間で入出力性能について評価を行った．この試験の結果，PnetCDF のインタフェースから Stampi の MPI-I/O インタフェースを通して入出力操作を行う場合，PnetCDF データに関連する情報を登録または検索する操作などにより，PnetCDF インタフェースを導入する事によるオーバーヘッドの影響ががデータ長が短い領域では大きかった．一方，データ長を大きくすると，その影響は小さくなっていった．このような大規模データであれば，複数の計算機で構成された環境で，計算機内と計算機間の違いや，データが存在する計算機のアーキテクチャの違いなどを意識する事なく netCDF のデータ形式で自由にシームレスな入出力操作が出来ることになり，有用であると考える．集団型入出力関数による入出力処理では，プロセス数の増加によりプロセス間の集団型データ通信などの処理が増えてゆき，処理全体に要する時間が増加するが，データ入出力のみに着目すれば，プロセス数の増大により，処理時間が短くなることが分かった．並列化されたプログラムにおいて非集団型入出力を用いる場合，計算プロセス個々にデータを配分したり，書き込みの為に一つのノードにデータを集めるなどの通信コストが余分にかかり，非集団型入出力を行うよりも集団型入出力を用いる方が有用であると考えられる．また，今回の性能評価から入出力操作において，ファイルをオープンする操作が割と大きな割合で占め

られているので，この操作を極力少なくなるように並列計算プログラムを作れば，十分な入出力性能が発揮できると期待できる．

今後の課題としては，未実装の入出力インタフェースへの対応や計算プログラムへの適用を考えている．

謝辞 本研究を進めるにあたり，日本原子力研究開発機構 システム計算科学センター長の矢川 元基 氏には数々の支援をして頂きました．また日本原子力研究開発機構 システム計算科学センターの皆様にはStampi のライブラリを提供して下さり，数々のアドバイスを頂きました．ここに感謝致します．

なお，本研究の一部は文部科学省 科学研究費補助金（若手研究（B）課題番号 18700074）および財団法人 カシオ科学振興財団 平成 16 年度研究助成により行われました．

参 考 文 献

1) Message Passing Interface Forum: *MPI: A Message-Passing Interface Standard* (1995).

2) Message Passing Interface Forum: *MPI-2: Extensions to the Message-Passing Interface* (1997).

3) Tsujita, Y., Imamura, T., Takemiya, H. and Yamagishi, N.: Stampi-I/O: A Flexible Parallel-I/O Library for Heterogeneous Computing Environment, *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, Lecture Notes in Computer Science, Vol.2474, Springer, pp.288–295 (2002).

4) Imamura, T., Tsujita, Y., Koide, H. and Takemiya, H.: An Architecture of Stampi: MPI Library on a Cluster of Parallel Computers, *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, Lecture Notes in Computer Science, Vol.1908, Springer, pp. 200–207 (2000).

5) Rew, R. K. and Davis, G. P.: The Unidata netCDF: Software for Scientific Data Access, *Sixth International Conference on Interactive Information and Processing Systems for Meteorology, Oceanography, and Hydrology*, American Meteorology Society, pp.33–40 (1990).

6) Li, J., Liao, W.-K., Choudhary, A., Ross, R., Thakur, R., Gropp, W., Latham, R., Siegel, A., Gallagher, B. and Zingale, M.: Parallel netCDF: A High-Performance Scientific I/O Interface, *SC '03: Proceedings of the 2003 ACM/IEEE Conference on Supercomputing*, IEEE Computer Society, p.39 (2003).

7) Thakur, R., Gropp, W. and Lusk, E.: On Implementing MPI-IO Portably and with High Performance, *Proceedings of the Sixth Workshop on Input/Output in Parallel and Distributed Systems*, pp.23–32 (1999).

8) Latham, R., Miller, N., Ross, R. and Carns, P.: A Next-Generation Parallel File System for Linux Clusters, *LinuxWorld*, Vol. 2, No. 1 (2004).

9) Thakur, R., Gropp, W. and Lusk, E.: An Abstract-Device Interface for Implementing Portable Parallel-I/O Interfaces, *Proceedings of the Sixth Symposium on the Frontiers of Massively Parallel Computation*, pp. 180–187 (1996).

10) Gropp, W., Lusk, E., Doss, N. and Skjellum, A.: A high-performance, portable implementation of the MPI Message-Passing Interface standard, *Parallel Computing*, Vol. 22, No. 6, pp.789–828 (1996).

11) Lee, J., Ma, X., Ross, R., Thakur, R. and Winslett, M.: RFS: Efficient and Flexible Remote File Access for MPI-IO, *Proceedings of the 6th IEEE International Conference on Cluster Computing (CLUSTER 2004)*, IEEE Computer Society, pp.71–81 (2004).

12) Gabriel, E., Resch, M., Beisel, T. and Keller, R.: Distributed Computing in a Heterogeneous Computing Environment, *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, Lecture Notes in Computer Science, Vol.1497, Springer, pp.180–187 (1998).

13) GridMPI: `http://www.gridmpi.org/`.

14) 石川 裕：YAMPII もう一つの MPI 実装，情報処理学会 研究報告，2004-HPC-99, Vol.2004, No.81, pp.115–120 (2004).

15) George, W.L., Hagedorn, J.G. and Devaney, J.E.: IMPI: Making MPI Interoperable, *Journal of Research of the National Institute of Standands and Technology*, Vol.105, No.3, pp. 343–428 (2000).

16) The National Center for Supercomputing Applications: `http://hdf.ncsa.uiuc.edu/HDF5/`.

17) Carns, P.H., Ligon III, W.B., Ross, R.B. and Thakur, R.: PVFS: A Parallel File System for Linux Clusters, *Proceedings of the 4th Annual Linux Showcase and Conference*, USENIX Association, pp.317–327 (2000).

18) Ross, R., Nurmi, D., Cheng, A. and Zingale, M.: A Case Study in Application I/O on Linux Clusters, *SC '01: Proceedings of the 2001 ACM/IEEE Conference on Supercomputing (CDROM)*, ACM Press, p.11 (2001).