近畿大学工学部研究報告 №39, 2005年, pp.140-147
Research Reports of the School of Engineering,
Kinki University №39, 2005, pp.140-147

PVFSを用いた計算機間MPI-I/O機能の最適化

辻田祐一[1]

Optimization in Remote MPI-I/O Operations Using a Parallel Virtual File System

Yuichi TSUJITA

abstract

MPI (Message Passing Interface) is the de facto standard in parallel computation and many parallel computer vendors have provided own MPI library. Although a vendor-supplied MPI library provides high performance MPI-I/O operations inside a computer, such operations among computers which have different MPI libraries have not been provided yet. To realize seamless MPI-I/O operations among computers, Stampi-I/O was developed. To cope with recent data-intensive parallel computation, a parallel file system named PVFS was supported in Stampi. Although remote MPI-I/O operations to a PVFS file system were realized, its performance was not sufficient for data-intensive computation. To improve throughput of the operations, optimization of a circular buffer in the remote MPI-I/O operations has been carried out, and sufficient performance improvement has been observed.

Keywords: MPI, MPI-I/O, Stampi, Stampi-I/O, PVFS, circular buffer

1. はじめに

近年、計算機シミュレーションは、より現実に近い結果を得るために、計算プログラムも、扱うデータの大きさも大規模化の一途をたどっている。大規模な計算機シミュレーションを行う為の手法の一つとしてMessage Passing Interface (MPI) [1, 2] と呼ばれる並列計算における標準的な通信インタフェースが提案されてきており、様々なMPIライブラリが開発され、盛んに利用されている。MPIは下層レイヤにある通信メカニズムなどを考慮しなくても自由に一様なインタフェースで計算プロセス間のメッセージ通信が利用可能であり、様々な並列計算機やPCクラスタなどで利用できる。ただし、異なるMPIライブラリ間で通信を行うことは出来ない。この機能を実現するためにStampi [3] が開発された。StampiはMPIライブラリの上に仲介インタフェースを形成し、ユーザ・プログラムからの計算機内、計算機間MPI通信を、通信相手により動的に通信経路を選択し、MPI通信を実現する。

一方、近年の大規模データを扱う並列計算の為に、MPIの仕様で新たに定められた並列入出力インタフェースであるMPI-I/O [2] がある。MPI-I/Oは様々なMPIライブラリで実装されてきているが、異なるMPIライブラリ間では利用できない。この機能を実現するために、Stampiにおいて計算機間でMPI-I/O機能を実現する機能が開発された [4]。さらに近年では様々な並列ファイルシステムが開発されてきており、その一つとしてLinuxが稼動するPCクラスタにおいて無償で構築できるParallel Virtual File System (PVFS) [5] がある。PVFSはPCクラスタの各ノードのディスク領域をネットワークを介して仮想的に一つのファイルシステムとしてユーザに提供するものである。このPVFSをStampiの計算機間入出力機能で利用できるような機能の拡張を行ってきた。本研究ではこの入出力操作において、性能

[1]近畿大学工学部電子情報工学科
Department of Electronic Engineering and Computer Science
School of Engineering, Kinki University

向上の為の循環バッファのパラメタを変えながら、計算機間入出力性能の変化を調べ、最適なパラメタの調査を行った。以下、本研究で行った機能の拡張並びに性能評価について述べる。

2. PVFS を用いた計算機間 MPI-I/O 機能

Stampi のアーキテクチャを図 1 に示す。Stampi は各計算機のネイティブな MPI インタフェース、TCP/IP の通信インタフェース、ファイルシステムの上にユーザ・プロセスとの仲介インタフェースとして実装されている。さらに計算機間 MPI 通信並びに MPI-I/O 操作において、計算ノードが外部と直接通信できない場合、通信中継プロセス（router process）が外部と通信可能なノードに起動され、このプロセスを介してデータ通信が行なわれる。Stampi における計算機

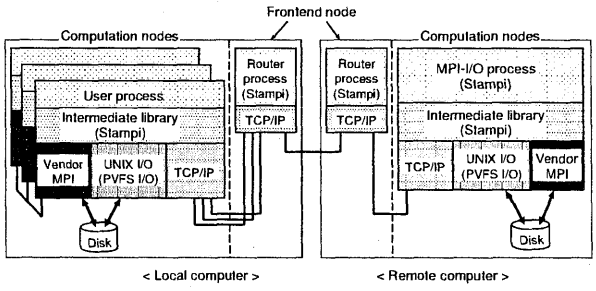

図 1: Stampi のアーキテクチャ。

間 MPI-I/O 機能は、まず、ターゲットの計算機上に入出力操作を行う MPI-I/O プロセスがリモートシェルコマンド（rsh 又は ssh）により起動される。MPI-I/O プロセスは、起動された計算機でベンダ提供の MPI-I/O ライブラリが利用できる場合、このライブラリを用いた高速な入出力を行うが、利用できない場合、UNIX I/O を用いた入出力を行う。さらに大規模なデータを効率よく扱うために、PVFS を用いた計算機間並列入出力機能を実現した。PVFS が利用可能な計算機に MPI-I/O プロセスが起動されると、UNIX I/O ではなく、PVFS が提供する PVFS I/O 関数を用いた入出力を行う。PVFS I/O 関数を利用することにより、UNIX I/O を用いるよりも高速な入出力操作が可能である。入出力を行うメカニズムについて、図 2 を用いて説明する。利用者はまず Stampi の起動

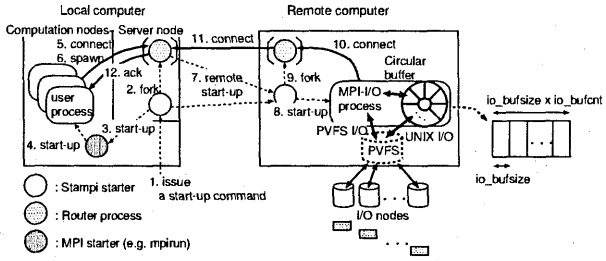

図 2: Stampi による計算機間入出力のメカニズム。

コマンドにより、Stampi の起動プロセス（starter）を起動する。この starter はネイティブな MPI ライブラリの起動プログラム（例えば mpirun）を起動し、この起動プログラムによりユーザ・プログラムが起動される。

ユーザ・プログラム内にはファイルをオープンするための `MPI_File_open()` があり、これが呼び出されると、事前に `MPI_Info_set()` により `info` オブジェクト内に設定されたリモート計算機のホスト名、作業ディレクトリ等のパラメタに基づき、リモート計算機上に MPI-I/O プロセスを起動する動作に入る。ここではまずリモート計算機上に Stampi の starter が起動される。すると starter はリモート計算機上に MPI-I/O プロセスを立ち上げ、MPI-I/O プロセスは指定されたファイルをオープンする。また必要に応じ、通信中継プロセスが外部と通信可能なノードに起動される。以上の操作により、リモート計算機との読み出し並びに書き込み関数が利用可能になる。入出力操作の完了後、ユーザ・プロセス内部で `MPI_File_close()` が呼び出され、MPI-I/O プロセスが当該ファイルを閉じ、MPI-I/O プロセスが消滅する。

Stampi は様々な MPI-I/O 関数をサポートしているが、その例として、`MPI_File_write()` 並びに

MPI_File_iwrite() の動作メカニズムを図 3(a)、(b) を用いて説明する。ユーザ・プロセスが図 3(a) にあ

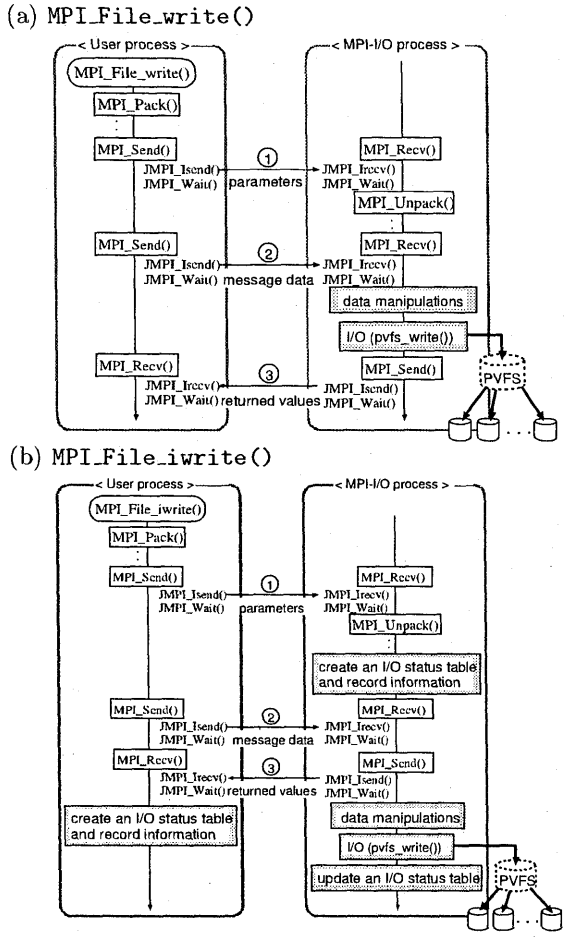

図 3: PVFS I/O 関数を利用した Stampi におけるリモート計算機への入出力メカニズム。

る MPI_File_write() を呼び出すと、Stampi の当該関数のインタフェース・ライブラリが呼び出される。その後、入出力に必要なパラメタ（関数名、ファイル名、データサイズ等）が MPI_Pack() により送信バッファに格納され、MPI-I/O プロセスに送られる。ユーザ・プロセス、MPI-I/O プロセス間のメッセージ通信は Stampi の MPI_Send() 並びに MPI_Recv() により行なわれる。これらの関数の内部では、Stampi が持つ通信関数 JMPI_Send()、JMPI_Recv()、並びに JMPI_Wait() による非同期な TCP ソケット通信が行なわれる。MPI-I/O プロセスは受け取った I/O 要求や関連するパラメタを取り出し、その I/O 要求に従った入出力操作を行う。この入出力操作では、MPI-I/O プロセスは、UNIX I/O 関数である write() の代わりに PVFS I/O 関数である pvfs_write() を用いる。ユーザ・プロセスは MPI-I/O プロセスが入出力操作を完了し、MPI-I/O プロセスから完了通知を受けるまで、処理がブロックされる。

一方、図 3(b) に示す MPI_File_iwrite() では、MPI-I/O プロセスに I/O 要求や関連するパラメタを送るところまでは同じであるが、MPI-I/O プロセスがそれらを受け取った後、MPI-I/O プロセスはそれらを一時的に管理テーブルに保管する。このメカニズムによりユーザ・プロセスは MPI-I/O プロセスの入出力操作の完了を待たずに次の処理に移ることが出来る。一方、MPI-I/O プロセスは受け付けた I/O 要求をテーブルから取り出し、その I/O 要求に従った入出力操作を行う。入出力完了後は、入出力に関連するパラメタを入出力用のテーブルに保持する。ユーザ・プロセスから入出力完了の確認の関数（この場合、MPI_Wait()）が発行されると、当該テーブルが参照され、入出力完了の確認が行われる。

3. 性能評価

本研究で開発した入出力機能をネットワークで繋いだ PC クラスタ間で性能評価を行った。それぞれの PC クラスタの仕様を表 に示すと共に、試験環境のセットアップを図 4 に示す。この試験で用いた 2

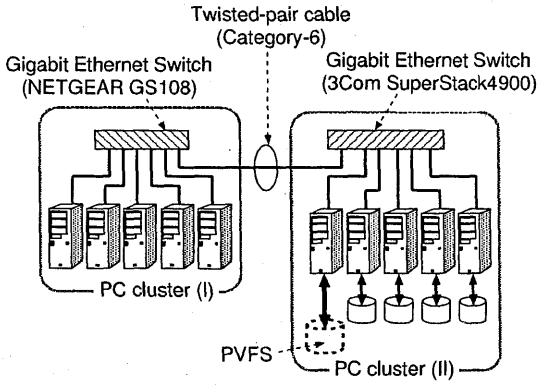

図 4: 性能測定のためのセットアップ。

つの PC クラスタは共に 1 台のサーバ・ノードと 4 台の計算ノードを持っている。ノード間は PC クラスタ I、II それぞれについて、NETGEAR GS108、3Com SuperStack3 4900 を用いて 1 Gbps の帯域で繋がれた。計算機間のデータ通信を最適化させるために、Stampi の起動コマンドのオプションにより、TCP ソケットの TCP_NODELAY オプションを有効にした。

表 1: 使用した 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 PowerEdge1600SC × 1	Dell PowerEdge1600SC × 1
	計算ノード	Dell PowerEdge600SC × 4	Dell PowerEdge1600SC × 4
CPU	管理ノード	Intel Xeon 2.4 GHz × 1	Intel Xeon 2.4 GHz × 2
	計算ノード	Intel Pentium-4 2.4 GHz × 1	Intel Xeon 2.4 GHz × 2
チップセット		ServerWorks GC-SL	
メモリ		1 Gbyte DDR 266 SDRAM	2 Gbyte DDR 266 SDRAM
ディスク装置	管理ノード	73 Gbyte (Ultra320 SCSI) × 1	
	計算ノード	40 Gbyte (ATA100 IDE) × 1	73 Gbyte (Ultra320 SCSI) × 2
イーサネットインタフェース	管理ノード	Intel PRO/1000-MT (オンボード)	Intel PRO/1000-XT (PCI-X ボード)
	計算ノード	Intel PRO/1000-MT (PCI-X ボード)	
イーサネットスイッチ		NETGEAR GS108	3Com SuperStack3 Switch 4900
Linux カーネル	管理ノード	2.4.20-28.7smp	
	計算ノード	2.4.21-2SCORE	2.4.21-2SCOREsmp
イーサネットドライバ	全ノード	Intel e1000 version 5.4.11	Intel e1000 version 5.5.4
MPI ライブラリ		MPICH-SCore (MPICH version 1.2.5 ベース)	

まず、PC クラスタ間の TCP ソケットによるデータ通信の性能を計測した。その結果を図 5 に示す。こ

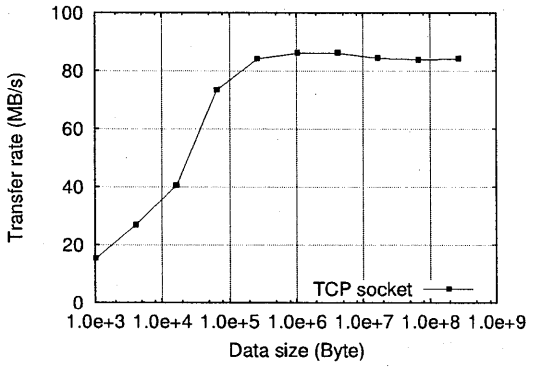

図 5: PC クラスタ間の TCP ソケット通信の性能。

の図に示すように、データ長が 512 KByte を超えるあたりからほぼ一定の性能 (~85 MB/s) が出ているが、理論帯域の約 68 %とあまり良くない。それぞれの PC クラスタで個別にノード間の TCP ソケット通信の性能を調べたところ、PC クラスタ II のノード間の性能は約 112 MB/s であったのに対し、PC クラスタ I の方は約 82 MB/s と悪かった。また、PC クラスタ I のスイッチを変更しても同様であった。よって PC クラスタ I のノードのネットワークインタフェース (関連するドライバソフトも含む) の性能が悪いことが分かった。以下の計算機間入出力試験では、この値を基準に評価を行った。

PC クラスタ I で Stampi の起動コマンドにより MPI プロセスを立ち上げ、ここから PC クラスタ II の PVFS ファイルシステムへの入出力操作の性能を計測した。まず集団型入出力関数の性能計測を行った。循環バッファの段数 (`io_bufcnt`) をデオルト値の 32 とし、各バッファのサイズ (`io_bufsize`) を変えて計測を行った。その結果を図 6 に示す。この結果から、`io_bufsize` に 2 MByte を設定した場合、最も性能が良くなり、読み出し操作ではデータ長が 4 MByte、書き込み操作では、64 MByte までは PVFS I/O を用いた場合よりも性能が良かった。しかしそれ以上のデータ長になると PVFS I/O を用いた場合の方が高い性能を示していた。これは、データ長が大きくなると循環バッファにデータが入りきらなくなり、PVFS I/O に比べ UNIX I/O による入出力性能が劣る点の影響が出てきてしまったためである。

次に、`io_bufsize` を 2 MByte に固定し、`io_bufcnt` を変えながら計測を行った。この条件において計測した集団入出力の性能を図 7 に示す。この図から、`io_bufcnt` に 128 を設定した場合、読み出し操作、書き込み操作共に PVFS I/O を用いる場合に比べ高い性能を示していた。この設定では、循環バッファにデータが全て入ってしまうため、殆ど計算機間の通信性能が全体の性能を決める大きな要因になっていた。特に読み出し操作では、この時の最大の入出力性能は計算機間のデータ通信性能とほぼ同じであった。

4. 関連研究

MPI の仕様に基づいた並列入出力機能を実現する試みはいくつかなされてきており、その代表として、

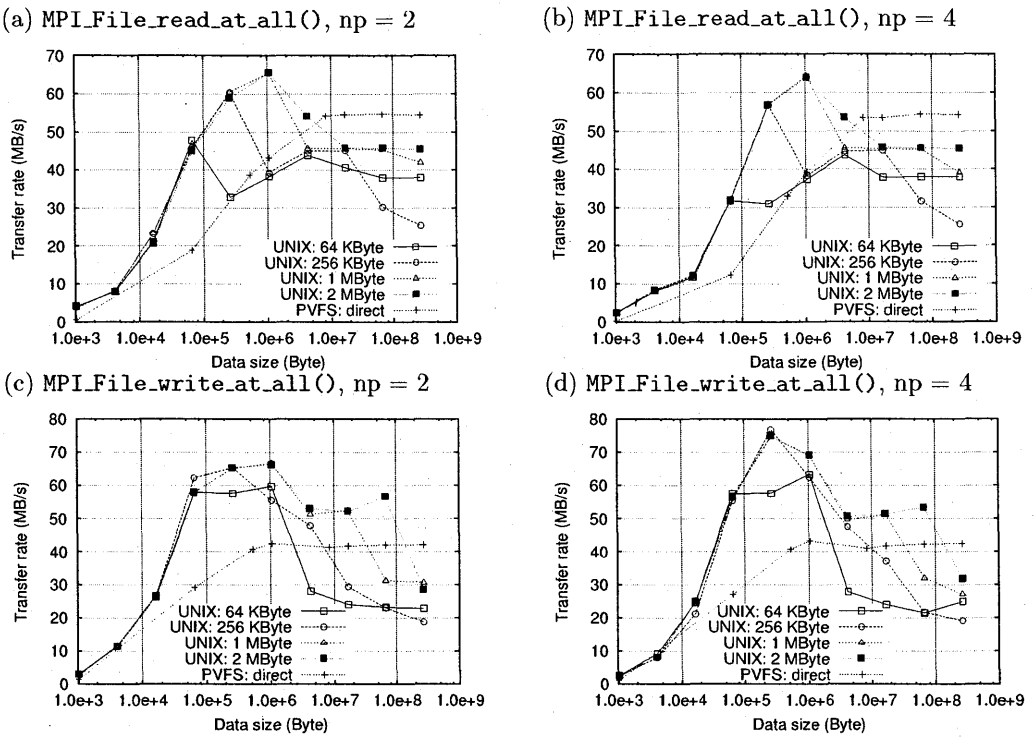

図 6: 集団型入出力の性能（io_bufcnt = 32）。

ROMIO [6] がある。ROMIO は、MPI ライブラリの模範的な実装として開発された MPICH [7] の MPI-I/O ライブラリとして提供されている。ROMIO における入出力では、ADIO [8] という様々なファイルシステムへの透過的な入出力インタフェースを提供するライブラリを通して入出力を行うため、ADIO の下層レイヤにあるファイルシステムを意識する事無く、MPI-I/O のインタフェースで入出力が可能である。さらに PVFS への入出力機能も実装されているが、異なる MPI ライブラリ間では利用出来ない。一方、Stampi は異なる MPI ライブラリ間の透過的な MPI 操作を実現するために開発されており、MPI-I/O 機能も計算機間で利用可能である。

5. まとめ

本研究では、PVFS を用いた大規模計算機間 MPI-I/O 機能の性能向上のために、入出力を行う MPI-I/O プロセスに循環バッファを実装し、性能向上のための最適化の試みを行った。

実装した循環バッファには、バッファの段数と各バッファのサイズをユーザ・プログラムから MPI の info オブジェクトに定めたパラメタに設定することによりバッファの構成を変える事ができる機能がある。本稿では、ネットワークで繋がれた 2 台の PC クラスタ間で、この各バッファのサイズ（io_bufsize）と段数（io_bufcnt）を変えながら転送性能を向上させる組合せを探す測定を行った。

まず段数をデフォルト値の 32 のままで、各バッファのサイズを変えながら測定を行ったところ、バッファサイズが 2 MByte で良い性能を示す事を確かめた。しかし、データ長が大きいところでは、PVFS I/O を直接用いる場合の性能よりは低かった。

次に、各バッファのサイズを 2 MByte に設定し、段数を変えながら計測を行ったところ、段数が 128 の時、データ長が 256 MByte まで高い性能を示す事を確認した。この時は、データ長が大きいところでも、PVFS I/O 関数を用いた場合よりも高い性能を示していた。

以上の測定により、循環バッファを実装したことは性能を向上させる事に有効であることを示した。今後の予定としては、この循環バッファを PVFS I/O 関数を用いる場合にも適用し、性能向上のための実装を行いたい。また、実際の計算アプリケーションに適用し、その性能を計測することも検討している。

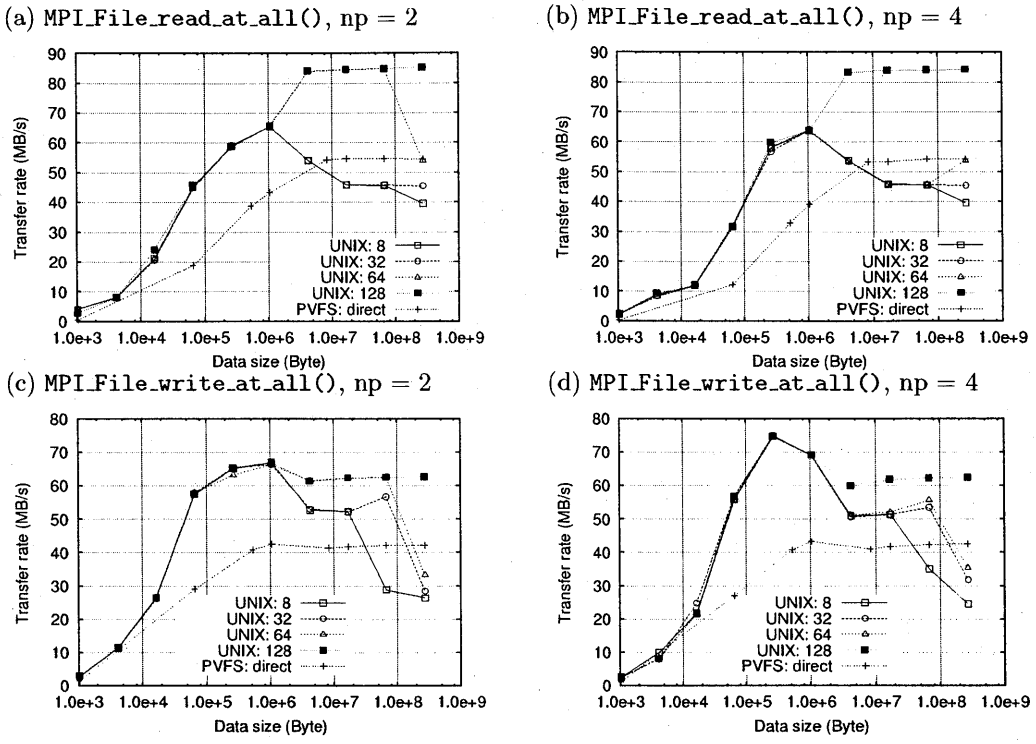

図 7: 集団型入出力の性能（io_bufsize = 2 MByte）。

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・センター長の矢川 元基 氏をはじめとする多くの方々には、本研究で用いた異機種計算機間通信ライブラリ Stampi を提供して頂き、また数々の御助言を頂きました。ここに感謝します。

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 15700079 並びに近畿大学 学内研究助成（奨励研究）課題番号 GS14 により行われました。

参考文献

[1] Message Passing Interface Forum. MPI: A message-passing interface standard. June 1995.

[2] Message Passing Interface Forum. MPI-2: Extensions to the Message-Passing Interface. July 1997.

[3] Toshiyuki Imamura, Yuichi Tsujita, Hiroshi Koide, and Hiroshi Takemiya. An architecture of Stampi: MPI library on a cluster of parallel computers. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface, 7th European PVM/MPI Users' Group Meeting*, volume 1908 of *Lecture Notes in Computer Science*, pages 200–207. Springer, 2000.

[4] Yuichi Tsujita, Toshiyuki Imamura, Hiroshi Takemiya, and Nobuhiro Yamagishi. Stampi-I/O: A flexible parallel-I/O library for heterogeneous computing environment. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, volume 2474 of *Lecture Notes in Computer Science*, pages 288–295. Springer, 2002.

[5] Philip H. Carns, Walter B. Ligon III, Robert B. Ross, and Rajeev Thakur. PVFS: A parallel file system for Linux clusters. In *Proceedings of the 4th Annual Linux Showcase and Conference*, pages 317–327. USENIX Association, October 2000.

[6] Rajeev Thakur, William Gropp, and Ewing Lusk. On implementing MPI-IO portably and

with high performance. In *Proceedings of the Sixth Workshop on Input/Output in Parallel and Distributed Systems*, pages 23–32, 1999.

[7] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A high-performance, portable implementation of the MPI Message-Passing Interface standard. *Parallel Computing*, 22(6):789–828, 1996.

[8] Rajeev Thakur, William Gropp, and Ewing Lusk. An abstract-device interface for implementing portable parallel-I/O interfaces. In *Proceedings of the Sixth Symposium on the Frontiers of Massively Parallel Computation*, pages 180–187, 1996.