ハイパフォーマンス
コンピューティング 84－8
（2000．12．8）

異機種計算機間通信ライブラリStampiにおける並列I/O機能の実装

辻田祐一[†] 今村俊幸[†]
武宮博[†,††] 山岸信寛[†]

我々は異機種並列計算機環境における分散並列I/Oライブラリとして MPI-2 の仕様に基づいた Stampi-I/O を Stampi の通信基盤を用い開発した。科学技術計算分野でのアプリケーションでは大規模なデータを扱う傾向にあり、そのような大規模データを効率的に管理する為に分散並列I/O機能が必要となってきた。このような状況に対し、複数の計算機間に効率的に分散して大規模データを管理するために、分散並列I/Oライブラリ Stampi-I/O を開発した。今回は開発の基礎となる機能のみに限定した開発及び性能評価を行なった。本稿では、Stampi-I/O を開発する目的、システムの構成、現状、性能評価テストそして将来の構築予定について述べる。

Parallel-I/O Implementation in Communication Library, Stampi, for Heterogeneous Computing Environment

YUICHI TSUJITA,[†] TOSHIYUKI IMAMURA,[†] HIROSHI TAKEMIYA[†,††]
and NOBUHIRO YAMAGISHI[†]

A distributed parallel-I/O library, Stampi-I/O, which is based on MPI-2, has been developed using communication infrastructure of Stampi. Many applications in this field handle huge amount of data. Distributed parallel-I/O system is required in effective handling of such huge data in heterogeneous computing environment. For effective distributed handling of such huge amount of data among several computers, Stampi-I/O has been developed. At the moment, R & D of primitive features of Stampi-I/O and performance measurement test has been done. In this report, motivations, architecture, preliminary results of performance measurement and future plan of Stampi-I/O are described.

1. はじめに

計算科学の分野では、多くのアプリケーションが大規模なデータを扱うようになってきている。このようなアプリケーションでは一台の並列計算機では扱えないような大規模なメモリ、ディスク容量そして計算能力が要求される。また一台の計算機では、効率的に大規模データを扱うためのメモリ、ディスク容量そして計算能力には物理的な制約からも限界がある。一方、ネットワーク上の複数の計算機やデータを有機的につないだ分散計算機環境を構築することにより、計算資源を仮想的に集約することができ、大規模な科学計算を行なう事が可能になる。しかし分散計算機環境においてプログラム開発および実行を行なう事は、各計算機の仕様の違いなどを考慮する必要があり、容易ではない。

このような状況に対し、我々は異機種の並列計算機により構築された分散計算機環境（異機種並列計算機環境）における並列計算を実現するために、MPI[1] および MPI-2[2] の仕様に基づいた異機種計算機間通信ライブラリ Stampi[3] を開発してきた。Stampi の開発目的は、MPI により記述された分散並列計算プログラムの異機種並列計算機環境における開発および実行をサポートすることである。

科学計算では大規模データを扱うケースが多く、効率的な大規模データの処理の為に、様々な並列I/Oシステムが提案され、利用されてきている。例えば、PANDA[4]、River[5]、そして VIPIOS[6] などがある。これらは総じてクラスタ・システムにおける並列I/Oシステムであり、各システムそれぞれにおいて性能向上のための様々な対策を行なっている。

分散計算機環境での並列計算において、大規模データがネットワーク上の複数の計算機に分散管理される

† 日本原子力研究所 計算科学技術推進センター
Center for Promotion of Computational Science and Engineering, Japan Atomic Energy Research Institute
†† 日立東北ソフトウエア（株）
Hitachi Tohoku Software, Ltd.

場合、リモートI/Oはローカル I/O と同様に重要である。MPICH[7] は様々な並列計算機において広く利用されている通信ライブラリ・パッケージである。この中で ROMIO[8] は MPICH での MPI-I/O の機能を担っている。リモート I/O を行なう際は、ROMIO は RIO[9] を用いて、クライアント・サーバ型の接続を形成し、リモート・ホスト上のデータにアクセスする。

一方、我々は異機種並列計算機環境において効率的な分散並列 I/O 機能を実現する為に、Stampi のライブラリ・パッケージの一部として Stampi-I/O を開発した。Stampi-I/O でも Stampi と同様に、動的プロセス生成メカニズムを用いた通信方式を採用している。この通信方式は上記のクライアント・サーバ型接続を行なう通信方式とは異なり、通信を行なう時のみ通信プロセスを生成し、通信を終了した後、通信プロセスを終了させる為、通信を行なわない時は計算機資源を無駄に使わずに済む。よって異機種並列計算機環境において各計算機資源の有効利用を行ないながらの I/O 操作が可能になる。

本稿では、Stampi-I/O の開発目的、機能、現時点での性能、そして将来の開発計画について述べる。

2. Stampi-I/O

科学計算アプリケーションが扱うような大規模データは、一台の計算機のメモリ、ディスク容量、および計算処理能力では効率的に扱えないものになっている。一方、ネットワーク上の計算機の CPU 処理能力やメモリ、ディスクを仮想的に結合して、全体として大規模な計算資源を有することで、上記のような大規模計算が可能になる。

大規模データを効率的にアクセスするには、ネットワーク上の複数の計算機に分散してデータを管理するという方法がある。そこで我々は、Stampi に複数の計算機にまたがる分散並列 I/O 機能（Stampi-I/O）を加えた。この機能により、大規模データを扱う科学計算アプリケーションにおいて、ネットワーク上の複数の計算機にデータを分散して効率的にアクセスできることが期待される。

図 1 に Stampi 及び Stampi-I/O の全体の機能図を示す。以下、Stampi および Stampi-I/O について説明する。

Stampi は計算機内部だけでなく計算機間の通信も可能にした通信ライブラリである。Stampi では、ある特定のノードのみ外部通信ができる計算機には図 1 に示すように、ルーター・プロセスをそこに走らせて間接通信を行ない、全てのノードが外部通信可能なら直接

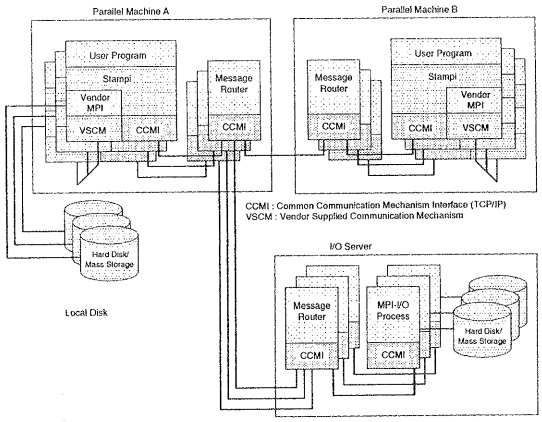

図 1 Stampi および Stampi-I/O の機能図。

通信を行なうような通信機構を実現している。Stampi を利用する事により、異なる通信機構を持つ計算機で構成された異機種並列計算機環境において、計算機間の通信を確立できる。よって異機種並列計算機環境においても分散並列計算を実現する事ができる。

通信を行なう相手が異なる並列計算機の場合、Stampi における共通通信プロトコル（Common Communication Mechanism Interface: CCMI）として採用されている IP（インターネット・プロトコル）を使用した通信が行なわれる。一方、同じ並列計算機内のプロセッサ間通信は、並列計算機に最適化されたベンダ提供の通信ライブラリ（Vendor Supplied Communication Mechanism: VSCM）を使用する。両者の切替えは Stampi 内部で自動的に行なわれるため、ユーザーは通信相手を意識する事なくプログラム開発及び実行が行なえる。

次に Stampi-I/O について説明する。Stampi-I/O は Stampi のライブラリ・パッケージの一部として実装されており、異機種並列計算機環境において、MPI-2 が定める MPI-I/O の機能をローカル及びリモート I/O において実現している。Stampi-I/O においても、Stampi と同じ通信基盤を用いている。Stampi と同じ通信基盤を利用することにした理由としては、通信部分のインタフェースを共通化できることや多種多様な外部通信ネットワーク特性を持つ環境でも柔軟に対応できるからである。

MPI-I/O で定める機能は、通常のファイル I/O に加えて、ストライド・アクセスや一つのファイルを物理的に複数のファイルに分割して管理する機能、集団的 I/O 操作などがある[2]。

図 1 において、ユーザーのプログラムはマシン A で

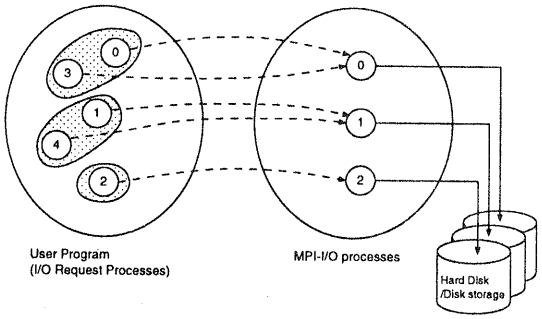

図 2　任意個の MPI-I/O プロセス起動の概念図。

表 1　Stampi-I/O において現在（2000 年 12 月）サポートされている関数。

関数の機能	サポートしている関数
ファイルのオープン、クローズ	MPI_File_open, MPI_File_close
ファイルの削除	MPI_File_delete
ファイルのサイズ変更	MPI_File_set_size
ファイル・ビュー関連	MPI_File_set_view, MPI_File_get_view
I/O 操作（ブロッキング/ノン・ブロッキング 集合的/非集合的）	MPI_File_read, MPI_File_write, 他 合計 20 個
ファイルのシーク機能	MPI_File_seek
I/O の同期	MPI_File_sync

動いている。この時、マシン A からローカルディスクへのアクセスがローカル I/O であり、マシン A から I/O サーバへのアクセスがリモート I/O である。MPI-I/O プロセスはリモートホストで I/O 操作を行なう時に起動される。

Stampi-I/O の実現方法であるが、まずローカル I/O の場合、ルーター・プロセスや MPI-I/O プロセスは起動されず、ベンダ提供 MPI-I/O ライブラリが利用できる場合はそれ自身が、利用できない場合は、Stampi が提供する MPI-I/O ライブラリが用いられる。一方、リモート I/O の場合、ターゲット・ホスト（I/O サーバ）上に MPI-I/O プロセスとルーター・プロセスが起動される。計算機間の通信は両計算機のルーター・プロセスを介して行なわれる。MPI-I/O プロセスがターゲット・ホスト上において I/O 操作を行なう時、ベンダ提供あるいは Stampi 提供の MPI-I/O ライブラリのどちらを用いるかはローカル I/O の場合と同様である。ローカル I/O とリモート I/O の切替えはターゲット・ホストにより Stampi-I/O 内部で自動的に切替えられる。

また Stampi-I/O では、**図 2** に示すように、任意個数の MPI-I/O プロセスの起動が行なえるようになっている。複数のユーザー・プロセスがリモート・ホスト上に I/O 操作を行なう場合、それぞれのプロセスに対応した MPI-I/O プロセスが起動される。この際起動する MPI-I/O プロセスの数がユーザー・プログラムのプロセス（I/O リクエスト・プロセス）の数よりも少ない場合、1 つの MPI-I/O プロセスが 2 つ以上の I/O リクエスト・プロセスからの I/O 要求を受け持つ。複数の I/O リクエスト・プロセスに対応付けられた MPI-I/O プロセスは、委託された I/O 操作を順番に実行する。この場合、各 MPI-I/O プロセスが受け持つデータサイズが小さくなる為、将来複数の通信経路がサポートされた時通信コストの面で有用である。

以上の特徴により、ユーザーは計算機間の通信仕様の違いを意識する事なく MPI-2 の仕様に基づいた I/O 操作が行なえる。現在、Stampi-I/O では**表 1** に示すような関数が利用できる。現在サポートしているプラットホームは、日立製作所 SR2201、SGI Onyx および富士通 VPP300 の 3 機種である。現時点ではこれらの I/O 関数において共有ファイル・ポインタはサポートしていない。

3. 性能評価

今回実装した Stampi-I/O に関し、ローカル及びリモート I/O でのブロッキング方式の書き込み操作に関して、我々の持つ異機種並列計算機環境である COMPACS[10] にある富士通 VPP300、日立製作所 SR2201、そして SGI Onyx（以下、それぞれ VPP、SR、Onyx と記す。）を用いて性能評価を行なった。上記 3 機種におけるネットワーク接続形態については、HIPPI（100 MB/s）が上記 3 つのそれぞれの計算機間、そして ATM（PVC, 21 Mbps）が SR、VPP 間で利用できる。

3.1 ローカル I/O

まずローカル I/O について VPP のベンダ提供 MPI-I/O ライブラリ[11] を用いてベンダ版起動方式と Stampi 版起動方式の二つについて性能評価を行なった。テスト環境の機能を説明したものを**図 3** に示す。この図で、ユーザー・プログラムから直接ベンダ提供 MPI ライブラリを呼び出しているのがベンダ版起動方式である。一方、ユーザー・プログラムから Stampi ライブラリを呼び出し、さらに Stampi ライブラリがベンダ提供 MPI ライブラリを呼び出しているのが、Stampi 版起動方式である。両者の違いはベンダ提供 MPI ライブラリの呼び出し方にある。

このテストで得られた結果を**図 4** に示す。この図

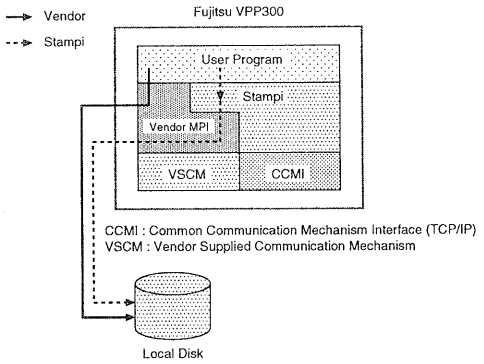

図 3　VPP300 でのローカル I/O 性能評価テストの機能図。

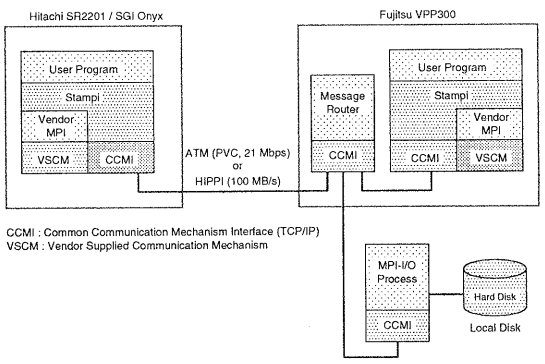

図 5　SR2201 又は SGI Onyx から VPP300 へのリモート I/O 性能評価テストの機能図。

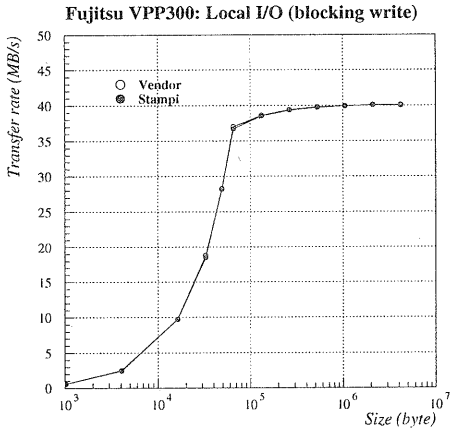

図 4　VPP300 でのローカル I/O の性能測定結果。

に示すように、このテストにおいてベンダ版起動方式と Stampi 版起動方式とでローカル I/O の性能には 4 MBytes のデータサイズまででは差が見られなかった。ゆえにこのデータサイズまででは、ベンダ版起動方式に比べて Stampi 起動方式による性能の損失はないと言える。

3.2　リモート I/O

リモート I/O の性能を評価するために、ローカル I/O の時と同様に VPP のベンダ提供 MPI-I/O ライブラリを用い、(1) SR から VPP と (2) Onyx から VPP の二つのケースについて VPP 上のディスクに対するブロッキング書き込み操作のテストを行なった。また比較のために、それぞれのケースについて計算機間の通信速度を計測した。通信速度は、相手の計算機にデータを送り、そのデータを受け取った相手計算機から送られてくるデータを受信するまでのラウンド・トリップの時間の半分を通信時間とし、通信速度を見積もった。

リモート I/O のテスト環境として、SR 又は Onyx から VPP へのリモート I/O について、図 5 に示すような環境で性能評価を行なった。SR および Onyx では各ノードが自由に外部と通信できるため、ルーター・プロセスは起動していない。一方 VPP では、特定のノードのみ外部と通信ができるため、そのノードにルーター・プロセスが起動している。

SR から VPP へのリモート I/O は以下のような環境で実現された。

- SR と VPP の間はユーザーが ATM あるいは HIPPI のどちらかを選択できる。
- データは Stampi の共通通信プロトコル (TCP/IP) を用い、VPP 上のルーター・プロセスを介して SR から VPP 上の MPI-I/O プロセスに送られ、この MPI-I/O プロセスにより VPP のディスクに書き込まれる。
- VPP ではベンダ提供 MPI-I/O ライブラリが利用出来るので、VPP 上の MPI-I/O プロセスはベンダ提供 MPI-I/O ライブラリを呼び出す。

このテストで得られた測定結果を図 6 に示す。この図に示してあるのは、SR と VPP の間の通信速度と SR から VPP へのリモート I/O の性能を ATM と HIPPI を用いた場合について、データサイズを変えながら計測した結果である。この図から分かるように、ATM を用いた場合、通信速度とリモート I/O の性能はほとんど変わらない。つまりローカル I/O の性能に比べて通信の性能がかなり低いために、リモート I/O の性能はほとんど通信速度によって決定されてしまっていたということになる。

一方 HIPPI を用いた場合、測定されたリモート I/O

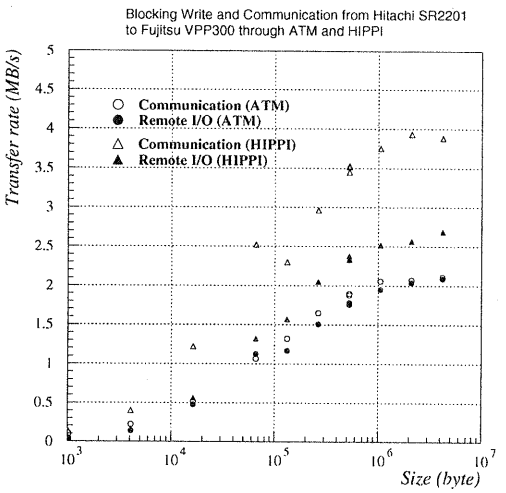

図 6 SR2201 から VPP300 への ATM または HIPPI を用いた時のリモート I/O の性能。

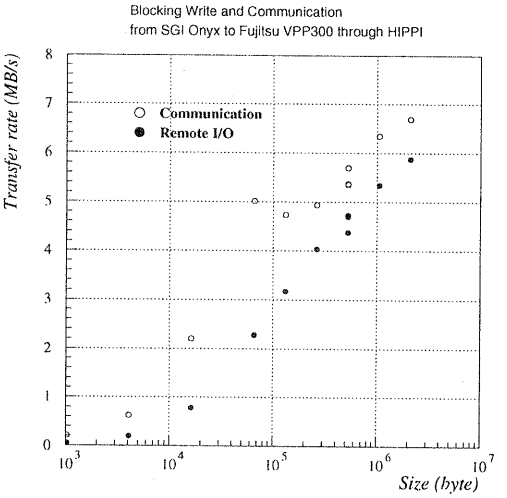

図 7 SGI Onyx から VPP300 への HIPPI を用いた時のリモート I/O の性能。

の性能は 1 MByte のデータサイズで 2.5MB/s で、VPP のローカル I/O の性能（1 MByte のデータサイズで 40 MB/s： 図 4 参照）と HIPPI を用いた通信性能（1 MByte のデータサイズで 3.8 MB/s：図 6 参照）から予想される性能（1 MByte のデータサイズで、$1/(1/3.8 + 1/40) \simeq 3.5$ MB/s）よりも悪い結果が得られた。この理由は現在調査中である。

次に Onyx から VPP へのリモート I/O の測定結果について説明する。データの書き込みの手順は SR から VPP へ書き込む場合と同様である。Onyx の場合は、ATM は利用できないため、HIPPI についてのみテストを行なった。

このテストで得られた測定結果を図 7 に示す。測定されたリモート I/O の性能は、1 MByte のデータサイズでは 5.4 MB/s で、事前に測定した通信速度（1 MByte のデータサイズで 6.4 MB/s：図 7 参照）と VPP でのローカル I/O の性能（1 MByte のデータサイズで 40 MB/s： 図 4 参照）から推測されるリモート I/O の性能（1 MByte のデータサイズで、$1/(1/6.4 + 1/40) \simeq 5.5$ MB/s）とほぼ同じである。ゆえに得られた性能はほぼ妥当なものと言える。

4. 今後の開発予定

Stampi-I/O の今後の開発予定として、

- 異機種並列計算機間の分散並列 I/O
- 集団 I/O に有効な派生データ型のサポート
- 任意個のルーター・プロセス起動

を考えている。これらの詳細について以下述べてゆく。

異機種並列計算機間の分散並列 I/O：現時点では、ファイルを複数の計算機間に分散して I/O 操作を行なう機構を備えていないが、今後大規模なデータを扱うためには、効率的にファイルを分散して並列 I/O を行なう必要がある。そのためにも早期にこの機能を実現することが必要である。現時点では、分散したファイルの情報が記述されるメタファイルを用いることで、並列分散 I/O を実現しようと考えている。

集団 I/O に有効な派生データ型のサポート：並列 I/O における集団 I/O においては、様々なアクセス・パターンが利用される。このようなケースでは MPI で定める基本データ型だけでは不十分で、派生データ型の利用が不可欠である。大規模データを扱う場合、複数のプロセスが同じファイルに対し領域分割を行なってアクセスする場合がある。このようなケースにおける集団 I/O において特に重要で頻繁に用いられるものからサポートしてゆく予定である。

任意個のルーター・プロセス起動：現在使用している計算機環境では 1 台の計算機に 1 つの通信経路という仕様になっている。ここで複数の通信経路が利用できる場合、現在の Stampi で通信中継を行なっているルーター・プロセスを 1 個から複数個に増やす事で通信にかかる時間が短縮できることが期待できる。

任意個のルーター・プロセス起動方式案の概念図を図 8 に示す。この図に示すように、5 個の MPI プロセスがリモート・ホスト上の 3 個の MPI-I/O プロセ

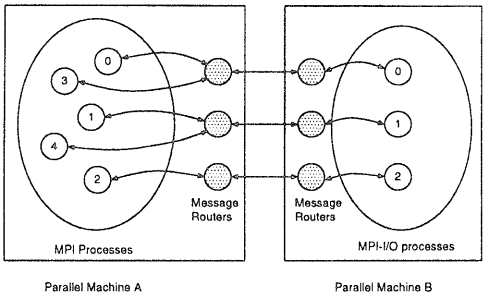

図 8　任意個のルーター・プロセス起動の概念図。

スを用いてリモート I/O を行なう際、通常ならば、それぞれのマシン上に 1 個ずつルーター・プロセスを起動して通信を行なうが、3 個のルーター・プロセスを起動することで 3 個の MPI-I/O プロセスとルーター・プロセス間は 1 対 1 の対応となり、MPI-I/O プロセス間はお互いに干渉することがなくなる。これにより効率的なリモート I/O の実現が期待される。

5. ま と め

我々は、異機種並列計算機環境において通信ライブラリ Stampi に加えて、分散並列 I/O 機能として Stampi-I/O を開発してきた。今回はブロッキング方式の書き込み操作について、富士通 VPP300 のベンダ提供 MPI-I/O ライブラリを用い、ローカル I/O の性能と日立製作所 SR2201 および SGI Onyx をユーザー・プログラムを実行するホスト、VPP300 を I/O サーバとした時のリモート I/O の性能の評価を行なった。性能評価を行なったほとんどのケースについて、計算機間の通信性能と VPP300 で得られたローカル I/O の性能から予測されるものに近いリモート I/O の性能が実測された。今回のテストは現状の環境で得られる基礎的な性能を得る事が目的であった。これらの結果を基にこれからの Stampi-I/O の開発を進めて行くつもりである。

また機能の充実だけでなく、実際にアプリケーションに適用することも考えて行きたい。

謝辞

High Performance Computing Center Stuttgart (HLRS) の U. Küster, M. Resch 両氏には Stampi-I/O の開発にあたり、様々な有意義なコメントを頂いた。ここに感謝の意を表したい。

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 Standard", July 1997.
3) T. Imamura, Y. Tsujita, H. Koide and H. Takemiya: "An Architecture of Stampi: MPI Library on a Cluster of Parallel Computers", In *Proceedings of 7th European PVM/MPI User's Group Meeting*, September 2000.
4) Y. Cho, M. Winslett, S. Kuo, J. Lee and Y. Chen: "Parallel I/O for Scientific Applications on Heterogeneous Clusters: A Resource-utilization Approach", In *Proceedings of the 13th ACM International Conference on Supercomputing*, June 1999.
5) R. Arpaci-Dusseau, E. Anderson, N. Treuhaft, D. Culler, J. Hellerstein, D. Patterson and K. Yelick: "Cluster I/O with River: Making the Fast Case Common", In *Proceedings of the 6th Workshop on I/O in Parallel and Distributed Systems*, May 1999.
6) P. Brezany, T. Mück and E. Schikuta: "A Software Architecture for Massively Parallel Input-Output", In *Proceedings of the Conference HPCN 97 Europe, Vienna, Austria*, pp. 811–820, April 1997.
7) W. Gropp and E. Lusk: "User's Guide for mpich, a Portable Implementation of MPI", (1998), `http://www-unix.mcs.anl.gov/mpi/mpich/`.
8) R. Thakur, W. Gropp and E. Lusk: "On Implementing MPI-IO Portably and with High Performance", In *Proceedings of the 6th Workshop on I/O in Parallel and Distributed Systems*, pp. 23–32, May 1999.
9) I. Foster, D. Kohr, Jr., R. Krishnaiyer and J. Mogill: "Remote I/O: Fast Access to Distant Storage", In *Proceedings of the 5th Workshop on Input/Output in Parallel and Distributed Systems*, pp. 14–25. ACM Press, November 1997.
10) 日本原子力研究所 計算科学技術推進センター ホームページ:http://guide.tokai.jaeri.go.jp/ccse/
11) Fujitsu UXP/V MPI 使用手引書 V11 用.