ハイパフォーマンス
コンピューティング 93－8
（２００３．３．１１）

PVFS-PMの実装と性能評価

瀬河浩司[†] 建部修見[†] 児玉祐悦[†]
清水敏行[††] 工藤知宏[†]

クラスタファイルシステムPVFSをPMv2通信スタック上に移植したPVFS-PMの実装と性能評価について述べる。PMv2はSCoreクラスタシステムが提供するスケーラブルで高性能な通信ライブラリである。PVFS-PMはTCP/IP上で動作するPVFSの通信部分をPMv2の記述で置き換えることにより実装した。PVFS-PMはGigabit Ethernetで構成された8I/Oノードのクラスタにおいて、TCPベースのPVFSに比べ、書き込みで1.07倍、読み込みで1.93倍の性能向上を達成した。

Implementation and Performance Evaluation of PVFS-PM

Koji SEGAWA ,[†] Osamu TATEBE ,[†] Yuetsu KODAMA ,[†]
Toshiyuki SHIMIZU [††] and Tomohiro KUDOH[†]

This paper discusses a design and implementation of *PVFS-PM*, which is a porting of the PVFS cluster file system to the PMv2 communication stack. PMv2 is a high performance communication library provided by the SCore cluster system software. The original TCP/IP based communication of PVFS is replaced with the PMv2 communication library. PVFS-PM improves the performance by factors of 1.07 and 1.93 for writing and reading, respectively, with 8 I/O nodes compared with the original PVFS on TCP on a Gigabit Ethernet connected SCore cluster.

1. はじめに

近年、コストパフォーマンスに優れたPCクラスタが多く用いられるようになってきている。これまでクラスタは高性能計算に主に用いられてきたが、クラスタファイルシステムを用いれば、大規模で高性能なファイルシステムをクラスタを用いて提供することができる。

現在フリーソフトウェアとして利用できるクラスタファイルシステムソフトウェアにPVFS[4]がある。PVFSはLinux上で動くオープンソースのクラスタファイルシステムである。クラスタファイルシステムの性能は、個々のノードのハードディスク性能、プロセッサの性能、ノード間の通信性能などに左右されるが、特にノード間通信性能の影響は大きい。PVFSはTCP/IP通信スタックの上に実装されているが、LinuxでのTCP/IP通信性能の問題により十分な性能が得られないことが明らかになっている[2]。

技術研究組合 新情報処理開発機構(RWCP)で開発されたクラスタシステムソフトウェアSCore[1]では、PMv2[3]と呼ばれる高性能低レベル通信ライブラリを提供している。PMv2はTCP/IPより軽量の通信プロトコルを用いており、実効性能が高い。

我々はPVFSをPMv2上に移植した。この論文では、従来のPVFSを"PVFS-TCP"と呼び、移植したものを"PVFS-PM"と呼ぶ。TCP/IPの代わりにPMv2通信ライブラリを使うことによって、より高い性能を実現できた。

SCoreはGigabitEthernetとMyrinetの2種類の通信ハードウェアに対して同一APIによりPMv2を提供しており、PVFS-PMは両方のネットワークで動作可能である。Gigabit Ethernet上ではPMv2とTCP/IPを同時に使用することができる。従って、PVFS-PMはTCP/IPを用いるプログラムと混在でき、TCP/IPを用いて通信する一般的なプログラムを実行するユーザも、追加するハードウェアなしにPVFS-PMの高い性能を利用することができる。

2. PVFSとSCore

2.1 PVFS

PVFSは、普及が進むLinuxクラスタのクラスタファイルシステムとして、クレムソン大学を中心に開発が進められているオープンソースのファイルシステムである。PVFSは、PCクラスタのそれぞれのノー

† 産業技術総合研究所
†† シナジェテック

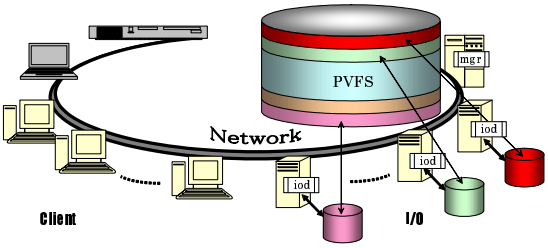

図 1 PVFS の構成

ドに分散しているローカルディスクを利用した並列ファイルシステムであり、ローカルディスクを提供するノード上で実行される I/O デーモン (iod)、ファイルシステムメタデータを保持するメタデータマネージャ(mgr) および、それらで構成される並列ファイルシステムへアクセスするためのクライアントライブラリで構成される (図 1)。

PVFS では、mgr および複数の iod により、ストライピングファイルシステムが構成される。それぞれのファイルは一定のストライプサイズごとに分割され、複数のディスクに格納される。

クライアントライブラリは、PVFS I/O API (PVFS native API) を提供する。クライアントライブラリでは、基本的にファイルオープン時にファイル情報を mgr から得たあと、ファイルへの書き込み読み込み時には mgr を介さず直接必要な iod と通信することにより、効率の良いファイル入出力を可能としている。また、PVFS I/O API により、ストライプサイズ、I/O ノードの数および I/O ノードのオフセット (ファイルのストライピングされた最初のブロックが格納されるノード) を指定することができる。

PVFS は、複数の iod と複数 disk を利用することにより、単一のファイルサーバを用いる NFS のようなネットワークボトルネックはなく、高バンド幅のファイル入出力を実現している。

PVFS では、Linux の VFS(仮想ファイルシステム) の実装もなされており、Linux の kernel module を追加することにより PVFS タイプのファイルシステムとしてマウントし、POSIX/Unix I/O API により PVFS を利用することができる。また、PVFS I/O API を利用した、ROMIO[10] の仮想 I/O デバイス adio も実装されているため、標準的な並列 I/O の API である MPI-2(MPI-IO) のインタフェースを利用することができる。

PVFS は基本的にユーザレベルの実装であり、カーネルの修正は必要ない。また、mgr、iod およびクライアントライブラリ間の通信はすべて標準的な TCP/IP を利用しているため、特定のメッセージパッシングライブラリに依存していない。

2.2 SCore

SCore クラスタシステムソフトウェアはワークステーションと PC クラスタのための高性能並列プログラミング環境を提供する。の並列分散システムソフトウェア研究室で開発され、現在は PC クラスタコンソーシアムでメンテナンスと配布が行われている [5]。

SCore は PMv2 とよばれる高性能な低レベル通信ライブラリを使っている。PMv2 については次節に記述してある。

3. PVFS-PM の実装

PVFS-PM の実装は、PVFS-1.5.5 を元にし、データ通信の部分を PMv2 2.1 API を用いて行った。

3.1 PMv2

PM は、クラスタコンピューティング用の低レベル通信ライブラリで、Myrinet や Ethernet などのクラスタで用いられる複数のネットワークや共有メモリを利用して通信する。TCP/IP に比べオーバーヘッドが小さい軽量で低遅延なプロトコル処理を実現するため、PM 通信プロトコルという独自プロトコルを実装し、カーネルトラップやカーネル空間とユーザ空間間のコピーもなくしている。また、一対一のメッセージパッシング API だけでなくリモートメモリ操作関数も提供している。現在提供されているバージョンは PMv2 と呼ばれる。

PMv2 は SCore ソフトウェア上でのマルチユーザ環境を実現するため、PMv2 チャネルと呼ばれる仮想ネットワークメカニズムを提供している。このチャネルは TCP/IP のようなコネクション指向の通信の代わりに信頼できるデータグラム通信を提供する。利用する全 PMv2 チャネルの状態の集合をコンテキストと呼ぶ。SCore ではプロセススケジュール時にこのコンテキストを保持することにより、複数の並列アプリケーションがそれぞれ PMv2 チャネルを自由に利用できる。

3.1.1 PVFS-PM

PMv2 を使ったプログラムはまず PMv2 デバイスを初期化する。これによりクラスタシステムで使われるノードのリストが得られる。そこで新しいコンテキストを開き、ノードとチャネルを関連付ける。最後にコンテキスト内における自身のノード番号を得る。

PMv2 はソケット API をサポートしていないため、ソケットレイヤーをエミュレートする必要がある。このため、以下のように `read()`、`write()` を `pmReceive()`、`pmSend()` で置き換える。

(1) ソケット API を PMv2 メッセージパッシング API で単純に置き換えるだけでは PVFS-PM の実装は不十分である。PVFS は接続のソースノード id を必要とするが、PMv2 のパケットヘッダには含まれていない。そこで、PVFS-

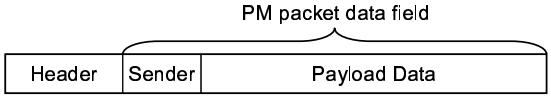

図 2 PVFS-PM パケットフォーマット

PM では各パケットのソースノード id、すなわち *Sender* をパケットデータフォーマットに追加し (図 2)、これをデータに追加/分離するコードを `write()`, `read()` の前後に追加する。

(2) PVFS-TCP においては、リモートからの I/O リクエストや ACK に `select()` や `poll()` システムコールを用いて複数の相手からのデータの到着待ちを行う。たとえば、iod は複数のクライアントからリクエストを受け、クライアントは複数の iod からのデータか ACK を待つ。`pmReceive()` は複数の sender から到着順にメッセージを受けるので、`select()`/`poll()` と `read()` の組は `pmReceive()` を用いて置き換えることができる。

(3) 一方、`write()` はそのまま `pmSend()` に置き換えることができる。

4. 性能評価

PVFS-TCP と PVFS-PM の性能を大きなファイルの読み書きバンド幅を測定することによって比較した。また、日常のファイルシステムとして PVFS が使用に耐えるかどうかを調べるために、ローカルディスク、NFS、PVFS-TCP、PVFS-PM のそれぞれのファイルシステム上に PVFS-TCP(PVFS-1.5.5) のソースファイルを展開し、make するのにかかる時間を測定した。

4.1 性能評価環境

PVFS-TCP の性能は、Linux 2.4.18 と Linux2.4.19 で測定したが、2.4.19 を使った場合の性能の方が 2.4.18 の場合に比べて (特に書き込みにおいて) 高いので、本報告中では 2.4.19 でのデータのみを示した。PVFS-PM については、SCore 5.2 パッケージで使われている Linux-2.4.18-2SCore を使った。他の評価環境要素は、次の通りである (図 3)。

- ノード計算機
  - Fujitsu PRIMERGY L200 × 17 台
    chipset: ServerWorks HE SL
    CPU: Pentium III 1.13GHz (2CPU SMP だが 1CPU のみ実装)
    メモリ: 1GB
    HDD: Fujitsu Model: MAN3184MC x 2, SEAGATE Model: ST373307LC x 1 SCSI 接続
    * システムと転送データ領域は別ディスク
- ネットワーク

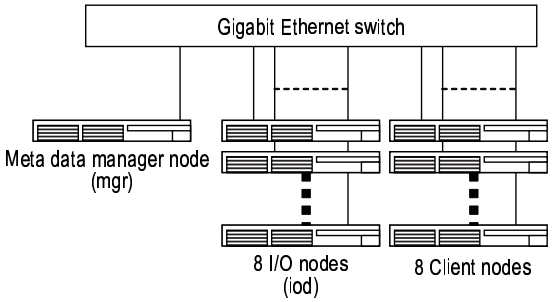

図 3 評価環境

  - NIC: 3Com 3C996B Gigabit Server NIC
  - Switch: 3Com SuperStack 3 (Switch 4924 3C17701)
- Gigabit Ethernet ドライバ
  - Broadcom Gigabit Ethernet Driver bcm5700 with Broadcom NIC Extension (NICE) ver. 2.0.18 (08/24/01)

4.2 バンド幅評価に用いたプログラム

PVFS クライアントライブラリを利用したディスク I/O ベンチマークプログラムを作成した。本ベンチマークプログラムでは、PVFS native API を用いて、クライアントマシンから PVFS ファイルシステムに対しファイルの読み書きを行う。プログラムのどこが性能ボトルネックとなっているか検証するために、プロセッサタイマを用いた正確な時間測定を用いた。このプログラムには、ブロックサイズ、全体のファイルサイズ、および PVFS のストライプサイズ、I/O ノードの数をパラメータとして渡すことができるようになっている。PVFS によるファイルの読み書きの処理の流れは次の通りである。まず、クライアントがブロックサイズ単位で pvfs_read()/pvfs_write() 関数を呼び出す。すると、PVFS ライブラリ関数がブロックサイズのアクセスをストライプサイズに分割して、設定ファイルに記述された順番に iod に割り当ててアクセスする。これをファイルサイズ／ブロックサイズだけ繰り返す。なお、iod、mgr はそれぞれ 1 プロセスが 1 ノードで動き、iod あるいは mgr だけが実行されている。以後 iod が動作しているノードを I/O ノードと呼ぶことにする。また、クライアントは 1 プロセス毎に別々のノードで実行される。

4.3 バンド幅の評価方法

前項に記述した環境を用いて、TCP および PM の場合についてそれぞれ block read/block write 性能の評価を行った。

ここで用いた評価プログラムの動作は以下の通りである。各クライアントとしては前項で述べたベンチマークプログラムを用いており、script による組み合わせで動作させている。多数のクライアントがそれぞれ個別の大規模ファイルをアクセスする。

図 4 PVFS-TCP Write 性能

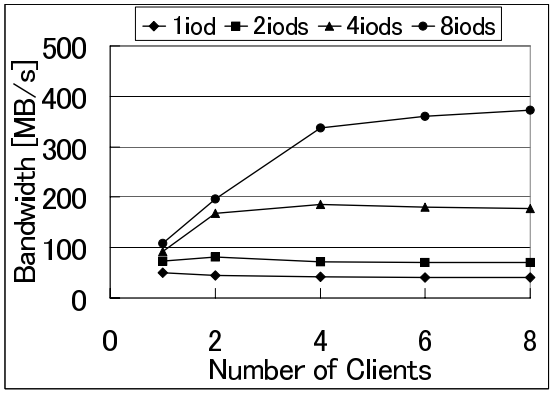

図 6 PVFS-PM Write 性能

図 5 PVFS-TCP Read 性能

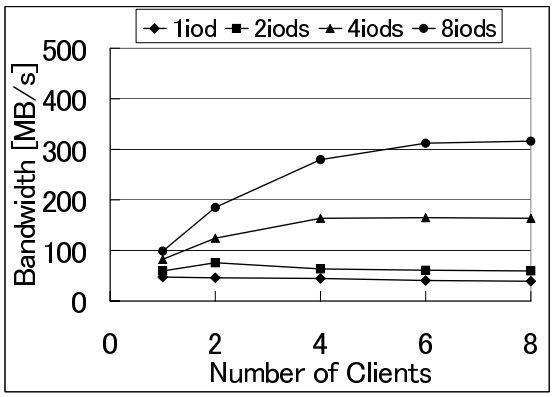

図 7 PVFS-PM Read 性能

(1) 全クライアントノードは一斉に同じ大きさの異なるファイルの書き込みを 3 回行う。このうち 2 回目の書き込みに要した時間を測定し、書き込みバンド幅を測定する。これは、書き込みに要する時間がクライアント毎に異なるため、終了が遅いプロセスがネットワークバンド幅を占有して使うことによる影響を避けるためである。

(2) 全プロセスはバリアにより同期する。

(3) 全クライアントノードは一斉に (1) で書き込んだファイルを 3 回読み出し、2 回目の読み出しに要した時間を測定して読み出しバンド幅を求める。3 回読み出しを行っているのは (1) と同じ理由による。

4.4 バンド幅の評価結果

PVFS-TCP および PVFS-PM の読み書きバンド幅は、図 4〜図 7 に示した。ファイルサイズは 2GB でストライプサイズは 64KB であり、示されているバンド幅は全てのクライアントの総バンド幅である。この測定においてそれぞれのクライアントのバンド幅はほぼ同じであった。PVFS-TCP の最大のバンド幅は、書き込みについては 349MB/sec (8 iod、8 クライアント)、読み込みについては 164MB/sec (8 iod、2 クライアント) であり、PVFS-PM の最大のバンド幅は、書き込みについては 373MB/sec (8 iod、8 クライアント)、読み込みについては 316MB/sec (8 iod、8 クライアント) であった。

書き込みバッファの影響を除けば、理論的な最大バンド幅は、ネットワークの総バンド幅とディスク I/O の総バンド幅のどちらか小さい方で制限される。GigabitEthernet のリンクバンド幅は、125MB/sec であり、ディスク I/O のバンド幅はそれぞれのノードで 50MB/etc 程度である。従って、ネットワークの総バンド幅は、

クライアント数 $\times$ $125MB/sec$

ディスク I/O の総バンド幅は、

iod の数 $\times$ $50MB/sec$

となる。

iod の数がクライアント数より小さい場合は、ネットワークバンド幅は iod の数に 125MB/sec をかけた値で制限される。しかしこの場合、ディスク I/O のバンド幅の方が小さく、全体のバンド幅はディスク I/O により制限される。図 8 は 8 iod の時の理論的な最大バンド幅と測定したバンド幅の比較である。理論的な最大バンド幅がオーバーヘッドを全く考慮せずに計算されていることを考えると、PVFS-PM はネットワークとディスクのバンド幅をほぼ有効に使っていること

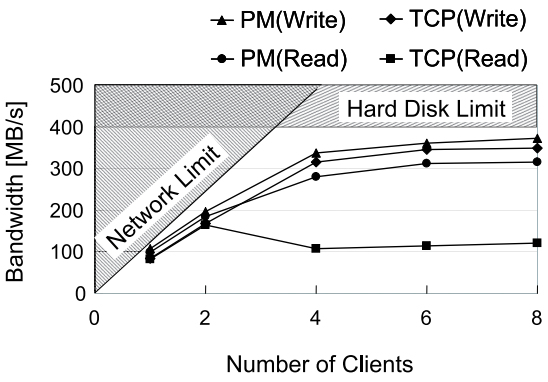

図 8 理論的最大値と測定値の比較

がわかる。PVFS-TCP の読み込みバンド幅はクライアントの数にスケールしていない。TCP のオーバーヘッドのみでは PVFS-TCP と PVFS-PM のバンド幅の大きな差を説明できない。PVFS-TCP のバンド幅低下の原因をつきとめるにはさらなる調査が必要である。

4.5 総合性能評価

実際にファイルシステムが使われる場合に近い条件での性能を見るため、local、NFS、PVFS-TCP(native)、PVFS-TCP(kernel module)、PVFS-PM(native)、PVFS-PM(kernel module) のそれぞれの場合について比較的小さくて多数のファイル I/O を行い、基本的なシステムコールにかかる時間をプロセッサタイマを用いて調べてみた。ここで、local とは local ディスクに対するファイル I/O。NFS は NFS サーバに対して NFS の 1 クライアントからのファイル I/O。PVFS に関しては、全て 1 iod、1 クライアントの構成でのファイル I/O であり、mgr、iod、クライアントは全て異なるホスト上で動作させた。

具体的にはまず、1MB のデータを書き込む処理について、異るファイルに対して `open()`、`write()`、`close()` を 1024 回繰り返す事により、それぞれにかかる時間の平均値を測定した。次に、1MB のデータを読み出す処理について、先ほど作成した 1024 個のファイルに対して `open()`、`read()`、`close()` を繰り返して、それらにかかる時間の平均値を測定した。それをグラフ化したものが図 9 である。

このグラフを見ると、PVFS-TCP と PVFS-PM の差はこのファイルサイズではほとんど見られない。

書き込み時については、PVFS-PM native は local と比べて `write()` が 3 倍程度遅くなっているのに対し、`open()` で 580 倍と非常に遅くなっている。NFS と比べても、それぞれ 2.6 倍と 23 倍遅い。読み出し時についてもほぼ同様である。

PVFS-PM の native と kernel module とを比較した場合、`write()` では kernel module が native より 1.7 倍遅い。これは kernel module のオーバヘッドといえる。一方、読み込み時の `open()` にかかる時間が native call を用いた場合に比べて 1/50 程度に小さくなっている。kernel module を利用した場合は，inode でファイルを管理しており、読み込みモードでファイルをオープンした場合、inode の内容は更新されたものである必要はなく、PVFS マネージャへのメタデータのアクセスだけとなっているためである。

5. 関 連 研 究

Vollestad[6] は、PVFS を SCI(Scalable Coherent Interface) に移植し、2 iod と 1 クライアントの環境で書き込みで 52.4MB/sec、読み込みで 29.9MB/sec の性能を得た。

Carns ら [4] は Myrinet[7] 上の TCP/IP を使い Chiba City クラスタ上で PVFS の性能評価を行った。この環境では、Myrinet 上での TCP/IP 通信性能が 37.7MB/sec しか得られておらず、8 iod での PVFS の読み書き総バンド幅は 180 から 255MB/sec である。32 iod と 28 クライアントでは 687MB/sec の読み込み総バンド幅を得ている。

Apon ら [8] は、Myrinet と GM メッセージパッシングシステムを使って、TCP/IP 上の PVFS の性能を評価を行った。7 iod の時に、260MB/sec の総バンド幅を得た。この評価では、iod とクライアントを同じノード上に割り振っているので我々の結果とそのまま比較することはできない。また、我々の PVFS-TCP の測定で現れたような性能低下現象は報告されていない。

Mache ら [9] は、GigabitEthernet で構成された 32 ノードのクラスタ上の PVFS で 1GB/sec の I/O スループットを達成したと報告しているが、ほとんどのディスクアクセスはローカルアクセスになるような環境を用いている。

6. 結　　論

我々は、SCore クラスタシステムソフトウェア上にクラスタファイルシステム PVFS-PM を実装した。大きなファイルの読み書きでは PVFS-PM では、8 iod までスケーラブルな性能が得られ、PVFS-TCP と比べて 8 iod での性能が書き込みにおいて 1.07 倍、読み込みにおいて 1.93 倍に向上した。PVFS-PM はユーザにネットワークとハードディスク性能をほとんど最大に利用可能とする高性能クラスタファイルシステムを提供するといえる。

一方、日常のファイルシステムとして用いるにはメタサーバの性能が不足しており、今後メタサーバの性能の向上が必要であると考えられる。

我々は、GigabitEthernet を使って PVFS の性能を評価した。さらに、PVFS の実装において計量通信ライブラリの適用の有効性を検証するため、Myrinet 上

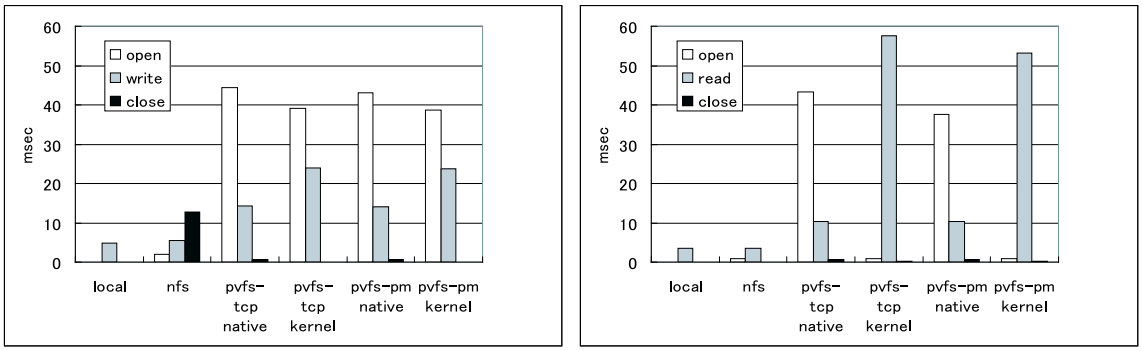

図 9 1M バイトのファイルの読み書き時間の内訳

で TCP と PMv2 を使い性能評価をする予定である。

PVFS-PM は PMv2 API で実装されているので、GigabitEthernet と Myrinet で構成されたクラスタ上で動かすことができる。

謝　　辞

本研究の機会を与えてくださった産業技術総合研究所グリッド研究センター長関口智嗣氏に感謝する。

参 考 文 献

1) Atsushi Hori, Hiroshi Tezuka, and Yutaka Ishikawa. User-level Parallel Operating System for Clustered Commodity Computers. In *Proceedings of Cluster Computing Conference 1997*, March 1997.

2) 瀬河, 建部, 児玉, 工藤. PVFS の性能評価とプラットフォームの検討. In 情報処理学会研究報告, HPC-91, pp.161–166, Aug. 2002.

3) S. Sumimoto. *A Study of High Performance Communication Using a Commodity Network of Parallel Computers*. PhD thesis, Keio, 2000.

4) P. H. Carns, W. B. Ligon III, R. B. Ross, and R. Thakur. PVFS: A Parallel File System For Linux Clusters. In *Proceedings of the 4th Annual Linux Showcase and Conference*, pages 317–327, Atlanta, GA, October 2000.

5) PC Cluster Consortium. *http://www.pccluster.org/*.

6) John Enok Vollestad. A high performance cluster file system using sci. In *Master's Thesis, Department of Informatics, University of Oslo*, 2002.

7) Myricom, Inc. *http://www.myri.com/*.

8) A. W. Apon, P. D. Wolinski, and G. M. Amerson. Sensitivity of Cluster File System Access to I/O Server Selection. In *Proceedings of CCGrid2002*, pages 183–192, May 2002.

9) J. Mache, J. Bower-Cooley, J. Guchereau, P. Thomas, and M. Wilkinson. How to achieve 1 gbyte/sec i/o throughput with commodity ide disks. In *Poster presentation of SC2001*, November 2001.

10) Rajeev Thakur,William Gropp, and Ewing . On Implementing MPI-IO Portably and with High Performance. In *Proceedings of the 6th Workshop on I/O in Parallel and Distributed Systems*, May 1999.