計算機アーキテクチャ 125−29
（1997. 8. 22）

ピンダウンキャッシュを用いたユーザレベルゼロコピー通信

手塚 宏史[†] 堀 敦史[†] Francis O'Carroll[††]
原田 浩[†††] 石川 裕[†]

メッセージ転送によって主記憶中の任意のアドレスのデータを転送する場合には，データコピーのオーバヘッドが発生し，データ転送バンド幅を制限する大きな要因となっている．我々は，このコピーオーバヘッドをなくし，データ転送バンド幅を高めるために，ユーザレベルのメモリ間ゼロコピー転送を実装した．さらに，そのために必要な物理メモリへのピンダウンのオーバヘッドを減らすために，ピンダウン領域を再利用するピンダウンキャッシュの手法を用いた．メモリ間ゼロコピー転送を MPI やソフトウェア分散共有メモリのプロトタイプに適用したところ，高いバンド幅や短いページ転送時間が得られることが示された．

A User level Zero-copy Communication using Pin-down cache

HIROSHI TEZUKA,[†] ATSUSHI HORI,[†] FRANCIS O'CARROLL,[††]
HIROSHI HARADA[†††] and YUTAKA ISHIKAWA[†]

When transferring data in the memory between nodes using a message transfer protocol, the data copy overhead limits the data transfer bandwidth. We have implemented a user level memory to memory zero-copy data transfer protocol to avoid the data copy overhead and to increase the data transfer bandwidth. We have also employed a pin-down cache method which reuses pinned-down area to decrease the overhead of pinning-down pages to physical memory. We adopted our memory to memory zero-copy data transfer and pin-down cache method to MPI and the prototype of software distributed shared memory, and obtained a high data transfer bandwidth and a low page transfer time.

1. はじめに

我々は，PC あるいは Unix ワークステーションを Myrinet[1] によって多数接続した PC/ワークステーションクラスタ[2),3)]上に，SCore と呼ぶギャングスケジューリングを用いたマルチユーザ並列プログラム実行環境を構築している[4)~6)]．Myrinet は 160M バイト/秒×双方向のリンクを持った高速ネットワークで，我々は Myrinet の性能を最大限に引き出すために，PM[7)]と呼ぶ通信ライブラリを独自に開発した．PM はユーザレベルのメッセージレイヤで，Pentium(166MHz) の PC 上で，7.5 マイクロ秒の低い通信レイテンシと 119M バイト/秒の高いバンド幅を得ている．しかし，PM は非同期のメッセージ転送をベースにしているため，主記憶中の大量のデータをそのまま別のノードに送るような場合にはバッファ領域との間のデータコピーが必要となり，それが実際のデータ転送バンド幅を制限していた．

本稿では，バンド幅の高いノード間通信を実現するためのメモリ間ゼロコピー転送，およびその実装に用いたピンダウンキャッシュついて報告する．まず，最初に 2 節で PM のアーキテクチャとその問題点について述べ，次に 3 節，4 節，5 節で，メモリ間ゼロコピー転送とピンダウンキャッシュおよびその実装について述べる．次に 6 節でメモリ間ゼロコピー転送とピンダウンキャッシュの評価について，7 節で関連研究について述べ，最後に 8 節でまとめと今後の課題について述べる．

2. PM のアーキテクチャと問題点

PM は Myrinet の高い性能をできる限り引き出すために開発したメッセージライブラリで，ユーザプロセスがネットワークハードウェアを直接ポーリングすることによってシステムコールや割り込みのオーバヘッドをなくし，低い通信レイテンシと高いバンド幅を得

† 技術研究組合 新情報処理開発機構 つくば研究センタ
並列分散システムソフトウェアつくば研究室
Parallel & Distributed System Software Laboratory
TRC,
Real World Computing Partnership
http://www.rwcp.or.jp
†† システム 21
††† 株式会社 SRA

ている．

PM は非同期のメッセージ転送をベースにしているため，ノード間通信は図 1 のように行なわれる．

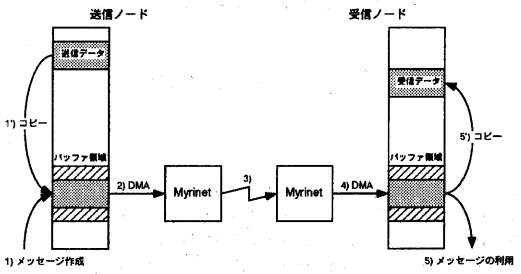

図 1　PM のアーキテクチャ

動的に生成される比較的短いメッセージの転送は，通常，次のような手順になる．

1) 送信ノードの CPU が主記憶中のバッファ領域にメッセージを作成する．
2) 主記憶中のバッファ領域から Myrinet インタフェースにメッセージを DMA 転送する．
3) ネットワークを介して，メッセージが受信ノードに送られる．
4) Myrinet インタフェースから主記憶中のバッファ領域にメッセージを DMA 転送する．
5) 受信ノードの CPU がメッセージを利用する．

このように主記憶中にバッファ領域を設けたのは，Myrinet インタフェースは DMA によって主記憶とデータ転送を行なうため，DMA を行なう領域は物理メモリにピンダウンされていなければならないためである．短いメッセージの場合には，どこかにメッセージを構築する領域が必要であり，特別なバッファ領域を設けることは大きなオーバヘッドにはならない．また，バッファ領域にメッセージが残るために，非同期通信が可能となり，フロー制御のためのメッセージの再送が容易に行なえる．

これに対して，PM のメッセージ転送によって，主記憶中の任意のアドレスのデータをそのまま転送する場合には，次のような手順になる．

1') 送信ノードの CPU が送信データをバッファ領域にコピーする．
2) 3) 4) 同上
5') 受信ノードの CPU がバッファ領域の受信データを必要なアドレスにコピーする．

このように，主記憶中の任意のアドレスのデータ転送の場合には，1') 5') のデータコピーが必要になり，データ転送のバンド幅を制限する大きな要因になる．図 2 に Pentium (166MHz) の PC 上でのデータコピーを伴う場合と伴わない場合の PM のデータ転送バンド幅を示す．このように，データコピーを行なわないバッファ領域からバッファ領域への場合には最大 119M バイト/秒 (8K バイトメッセージ) のバンド幅が得られているのに比べて，任意のアドレスからバッファ領域へのデータコピーを行なった場合には，PM のデータ転送バンド幅は 25.3M バイト/秒 (同上) と大きく減少している．これは，DMA を含んだネットワークのデータ転送バンド幅 (約 130M バイト/秒*) に比べて，主記憶中のデータコピーのバンド幅が小さいためである．

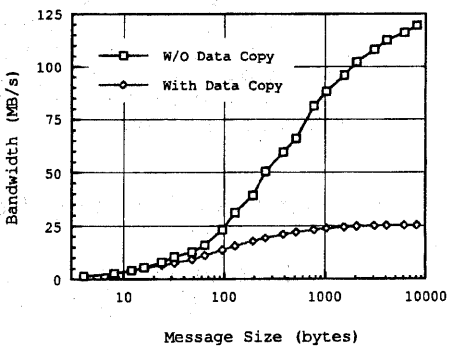

図 2　PM のデータ転送バンド幅

表 1 にいくつかの機種について主記憶中のデータコピーのバンド幅を実測したものを示す．この実験では，キャッシュの影響を排除するために 2 次キャッシュより大きな領域の転送時間を計測した．また，キャッシュラインのアラインメントの影響を調べるために，コピーするアドレスを少しずつずらしながらバンド幅の最大値と最小値を求めた．

表 1　データコピーバンド幅 (MB/s)

マシン	最小値	最大値
SS20/71	13.2	25.8
Ultra SPARC	188.0	205.5
Alpha PC	66.2	77.6
Pentium (166MHz)	31.0	31.5
Pentium Pro (200MHz)	52.3	48.7

このようにデータコピーのバンド幅は機種によって大きく異なる．データコピーによるオーバヘッドを減らす一つの方法は，Alpha や Ultra SPARC などのコピーバンド幅の高い機種を用いることであるが，我々は次に述べるユーザメモリ間のゼロコピー転送を PM に実装してこの問題を解決した．

* Myrinet のバンド幅は 160M バイト/秒であるが，PC の I/O バスである PCI の DMA バンド幅，約 130M バイト/秒によって制限される．

3. ユーザメモリ間ゼロコピー転送

ユーザメモリ間のゼロコピー転送は，図3のように Myrinet のハードウェアが主記憶中のデータを直接 DMA を用いて転送する．

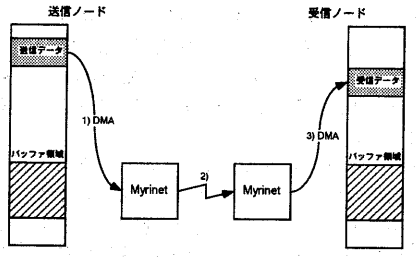

図3 ユーザメモリ間ゼロコピー転送

このようなメモリ間の直接転送を行なうには，送受信のデータ領域を物理メモリにピンダウンして，Myrinet インタフェースに対して物理アドレスを通知しておく必要がある．そのためには a) 受信ノードでは受信に先だってデータ領域を設定してデータ転送を行なう同期送受信プリミティブを用いる方法や，b) 送信ノードがアドレスを指定して受信ノードにデータを書き込む，リモートメモリアクセスによる方法が考えられるが，我々は後者のリモートメモリアクセスによる方法を選択した．その理由は，同期送受信プリミティブを用いる方法では，受信が完了するまで送信側が待たされるなどアプリケーションの自由度が低くなることと，リモートメモリアクセスではバッファの不足あるいは受信領域の未設定により発生するメッセージ喪失を防ぐための再送などのフロー制御が不要となり，制御が簡単になってそのためのオーバヘッドを小さくできると考えたからである．この反面，送信ノードは受信データ領域のページマップを知っている必要があるため，事前にマップ情報を送信ノードに送るネゴシエーションが必要となる．

リモートメモリアクセスによるデータ転送は次のように行なわれる．

1) 受信ノードの CPU は受信データ領域をピンダウンし，マップ情報を送信ノードに送る．
2) 送信ノードの CPU は受信ノードからマップ情報を受信する．
3) 送信ノードの CPU は送信データ領域をピンダウンする．
4) 送信ノードの CPU は送信側のデータアドレス，受信側のデータアドレスを指定してデータ転送を起動する．
5) 送信ノードの Myrinet インタフェースは DMA により主記憶からデータを読み出して送信する．
6) 受信ノードの Myrinet インタフェースは DMA によりデータを主記憶中のピンダウンされた領域に書き込む．
7) 受信ノードの CPU はピンダウンを解除する．
8) 送信ノードの CPU はピンダウンを解除する．

上記の手順のうち，ピンダウンおよびネゴシエーション 1) 2) 3) 7) 8) については同じ領域のデータを転送する場合には毎回繰り返す必要はない．

データ領域全てをピンダウンすることが可能であれば，上記の手順のうち 1) 2) 3) については一度だけ行なえば良いが，我々のシステムでは次のような理由からその方法は採用しなかった．まず，我々が目標としているマルチユーザの並列プログラム実行環境では複数のプロセスが時分割で動作しているため，一つのプロセスが大量の物理メモリを占有することは好ましくない．また，全データ領域のマップ情報を持たせるための大きなページテーブルが Myrinet インタフェース側に必要になる．

しかし，ピンダウンのオーバヘッドは比較的大きいため，データ転送毎に毎回ピンダウン/解除を繰り返していたのでは，ゼロコピー転送の利点であるバンド幅を高めることはできない．我々は，この問題を解決するために，次に述べるピンダウンキャッシュの手法を用いた．

4. ピンダウンキャッシュ

一度ピンダウンされた領域を解除せずに再利用すれば，実際のピンダウン操作を省くことができ，それに伴うオーバヘッドを減らすことができると考えられる．我々はこの手法をピンダウンキャッシュと呼んで PM のメモリ間ゼロコピー転送に実装した．ピンダウンキャッシュの動作は次のように行なわれる．

1) ピンダウン操作が行なわれた場合には，最初にすでにピンダウンされている領域を検索し，新しくピンダウンすべき領域を含む領域がすでにピンダウンされていればその領域を再利用し，ピンダウンは行なわない．
2) まだその領域がピンダウンされていない場合には，ピンダウンされた領域の合計がユーザプロセスに許可された最大値を超えたか，Myrinet インタフェース内のページテーブルのエントリが不足していないかを調べ，その場合には今後最も使われないと思われる領域のピンダウンを解除する．
3) 新しい領域をピンダウンする．
4) データ転送後のピンダウンの解除操作では，実際にピンダウンの解除はおこなわずに，その領域をピンダウン済みとして登録しておく．

このように，ピンダウンキャッシュはピンダウン領域の解除を実際に必要になるまで遅らせたものであ

る．データ転送を行なう領域より一度にピンダウン可能な領域が大きい場合にはピンダウンキャッシュは常にヒットし，ヒット確認のためのオーバヘッドを除けば固定的にピンダウンを行なう場合とほぼ同様な条件になる．

5. 実　　装

ピンダウンキャッシュを用いたメモリ間のゼロコピー通信は，次のようにユーザレベルのライブラリと，オペレーティングシステム・カーネル内のデバイスドライバ，および Myrinet インタフェース上のソフトウェアによって実装されている．

カーネル内デバイスドライバ
- ページのピンダウン/解除を行なう．
- Myrinet インタフェース上のページテーブルを設定する．

ユーザレベル・ライブラリ
- ピンダウン，転送開始などのアプリケーションインタフェースを提供する．
- ピンダウンキャッシュの管理を行なう．

Myrinet インタフェース上のソフトウェア
- ページテーブルの内容に基づいて主記憶とのDMA 転送を行なう．
- 他のノードとのデータの送受信を行なう．

ページのピンダウンは mlock/munlock システムコールを用いてユーザレベルで行なうこともできる*が，Myrinet インタフェース内のページテーブルの設定のためには物理ページ番号が必要であり，ユーザレベルでは物理ページ番号を取得する方法がないため，ピンダウンとページテーブルの設定の両方をまとめてカーネル内に実装した．これらをカーネル内に実装したもう一つの理由は，メモリ間データ転送では DMA のために物理アドレスを用いるため，ユーザレベルで物理アドレスを操作するのは危険だからである**．

現在の実装では，Myrinet インタフェース内のページテーブルエントリの数は 1024 個，ページサイズは 4k バイトなので，最大 4M バイトの領域を一度にピンダウンできるが，これは NetBSD でユーザがピンダウン可能な上限値よりも小さい．また，ピンダウン領域の管理には LRU を用いている．

* 一般の Unix ではピンダウンはスーパーユーザだけに許されているが，NetBSD ではある上限値までは一般ユーザでもピンダウンを行なうことができる．

** PC の I/O バスである PCI は一般に物理アドレスを用いて主記憶にアクセスするために Myrinet インタフェース内にページテーブルを持つ必要があるが，SBus のようにバスインタフェースがアドレス変換の機能を持っている場合には Myrinet インタフェースは物理アドレスを扱う必要はない．しかしこの場合でも，カーネルはバスインタフェースのアドレス変換情報を設定しなければならない．

6. 評　　価

本節では，ピンダウンのオーバヘッド，メモリ間ゼロコピー転送のバンド幅，および，現在開発中の SCore 上の MPI にゼロコピー転送を実装した場合の通信バンド幅，同じく開発中のソフトウェアによる分散共有メモリシステムのプロトタイプにゼロコピー通信を用いた場合のページ転送時間の実測結果を示す．これらの計測はすべて 160MB/s の Myrinet で接続した Pentium (166MHz) + NetBSD/i386 1.2 の PC を 2 台用いて行なった．

まず，ピンダウンキャッシュがヒットした場合とミスした場合について，ピンダウン操作にかかる時間の実測結果を表 2 に示す．この実験では，同じ領域のピンダウン/解除を 100000 回繰り返してその経過時間から 1 回の操作あたりの時間を計算した．また，ピンダウンキャッシュ・ミスの値は，強制的にキャッシュミスするようにキャッシュ管理ののソースコードを一部変更して計測した．

表 2　ピンダウンのオーバヘッド

ピンダウンキャッシュ	時間 (μ 秒)
ミス	90.8 + 12.3 × ページ数
ヒット	0.6

このように，ピンダウンキャッシュがミスした場合には，ネットワーク上のデータ転送時間 (25.6μ 秒/4K バイト) に比べて，ピンダウン操作のオーバヘッドは非常に大きい***と言える．

次に，図 4 にメモリ間ゼロコピー転送のバンド幅の実測結果を示す．これは，データ長を 4 から 8192 バイトまで変えながら，同じメモリ領域を 100000 回連続して転送して，その経過時間からバンド幅を求めたものである．受信ノードでは，転送に先だって一度だけピンダウンを行なって物理アドレスを送信ノードに通知し，送信ノードではデータ転送毎に毎回ピンダウン操作を呼び出している．さらにこの実験では，強制的にキャッシュミスするようにソースコードを一部変更して，ピンダウンキャッシュのヒット率を 0% から 100% まで変化させて計測した．

図 4 のバンド幅が全体的に図 2 に示したメッセージ転送 (コピーなし) の場合に比べて低下しているのは，たとえピンダウンキャッシュが 100% ヒットしていてもキャッシュ操作のためのオーバヘッドがあることに加えて，ゼロコピー転送では非同期のメッセージ転送と異なり，基本的に同期転送となるからである．

この実験では，ピンダウンキャッシュが 100% ヒッ

*** ピンダウンのオーバヘッドはプロセッサやオペレーティングシステムで大きく異なる．例えば Pentium Pro (200MHz) + NetBSD/i386 1.2 では約 26μ 秒である．

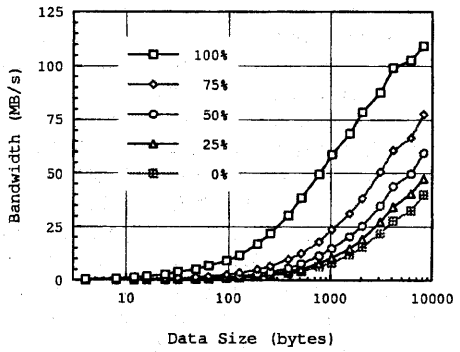

図4　メモリ間ゼロコピー転送のバンド幅

トしている場合には最大 109.4M バイト/秒 (データサイズ 8K バイト) のデータ転送バンド幅が得られた. ヒット率が 0% の場合でも最大 40M バイト/秒 (同上) が得られ, これはデータコピーを伴う場合のメッセージ転送の値よりも大きい.

実際のアプリケーションでは, ゼロコピー転送のバンド幅はピンダウンキャッシュのヒット率によって図4のグラフの範囲内の値になると考えられる. これらのことから, ピンダウンキャッシュのヒット率がゼロコピー転送の性能を左右する大きな要因であることがわかる.

次に, 現在我々が開発中の SCore 上の MPI[8),9)] にゼロコピー転送を適用した場合の同期 Send/Receive によるデータ転送バンド幅の計測結果を図5に示す. 比較のために, PM のメッセージ転送を用いた非同期 Send/Receive のバンド幅も同時に示した. なお, この実験では同じデータ領域を繰り返し転送しているためにピンダウンキャッシュは常にヒットしている.

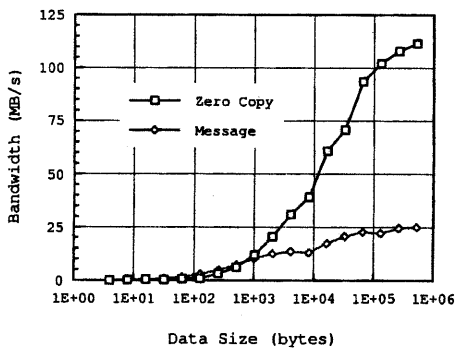

図5　MPI のデータ転送バンド幅

データコピーを伴うメッセージ転送を用いた場合には最大バンド幅は約 25M バイト/秒 (データサイズ 512K バイト) であったものが, ゼロコピー転送を用いることによって約 111.5M バイト/秒 (同上) と, 大幅に性能を改善することができた. 図5のゼロコピー転送を用いた場合のバンド幅が図4に示した値よりも全体的に小さいのは, MPI の場合にはデータ転送毎にピンダウン操作と送受信ノード間のネゴシエーションを行なっているからである. このように, ゼロコピー転送を用いることによって, 高いバンド幅を得ることができたが, 実際のアプリケーションに適用する場合には, 測定条件に述べたように, メッセージ転送とゼロコピー転送では別の API を用いていることに注意が必要である.

最後に, 表3に我々が開発中のソフトウェア分散共有メモリのプロトタイプにゼロコピー転送を適用した場合の, ページインに伴う 1 ページ (4K バイト) の転送時間を計測した結果を示す. 表3はページ転送時間だけの計測結果で, ページフォルトの処理時間などは含まれていない. なお, この実験では同じデータ領域を繰り返し転送しているためにピンダウンキャッシュは常にヒットしている.

表3　ソフトウェア分散共有メモリにおけるページ転送時間

方式	時間 (μ 秒)
メッセージ転送	385
ゼロコピー転送	110

メッセージ転送を用いたバージョンでは, 送信側, 受信側双方にバッファ領域とのページのコピーが必要であるが, ゼロコピー転送では直接ページの内容を転送するためにコピーが不要となり, ページ転送時間が大幅に短縮された.

さらに, ここには示していないが, メッセージ転送を使ったバージョンでは, 一旦ページの保護属性を書き込み可にしてデータコピーを行なった後に読み出しのみに変更する, などの処理が必要であるが, 物理ページに直接 DMA 転送を行なうゼロコピー転送ではそのようなページ保護属性の一時的な変更は不要であり, 全体的にはページインの時間がさらに短縮されると期待できる.

7. 関連研究

LHCP[10)] では, BIP(Basic Interface for Parallelism)Messages[11)] と呼ぶルーズ・ランデブのセマンティクスに基づいたメモリ間転送プロトコルを開発しており, Pentium Pro (200MHz) + Linux の PC 上で, 4.3 マイクロ秒 (データサイズ・ゼロバイト) の通信レイテンシと 126M バイト/秒 (データサイズ 8M バイト) の通信バンド幅を得ている. しかし, そのセマンティクスのため, 送信ノードでは先に送信したデータが受信されるまで, 次の送信を行なうことはできない.

Illinois 大学の FM(Fast Messages)1.1[12)] は, 送信

側の主記憶から Myrinet インタフェースへの転送を CPU が行なうことにより，DMA 可能な領域へのコピーオーバヘッドを削減しているが，通信バンド幅は CPU の I/O デバイスへの書き込みバンド幅で制限されている．

8. まとめと今後の予定

今回 PM に実装したメモリ間ゼロコピー転送によって，データコピーのオーバヘッドを省き，データ長が大きい場合には，ほぼハードウェア性能に近いデータ転送バンド幅を得ることができた．また，DMA を行なうために必要な物理ページへのピンダウンのオーバヘッドを減らすために，ピンダウンされた領域を再利用するピンダウンキャッシュを実装した．メモリ間ゼロコピー通信を MPI の同期通信やソフトウェア分散共有メモリに適用した結果，これらの手法が性能を高める上で有効であることが示された．

今回の実験では，実際のアプリケーションでのピンダウンキャッシュのヒット率や，それがアプリケーションの実行速度に与える影響については計測できなかった．現在開発している MPI およびソフトウェア分散共有メモリを用いて実際のアプリケーションを実行し，ピンダウンキャッシュおよびキャッシュの管理アルゴリズムの有効性の検証，ピンダウン領域の大きさの制限などによる，ヒット率の変化などについての計測を行ない，PM のさらなる改良を行なうことが今後の課題である．

参 考 文 献

1) http://www.myri.com.
2) 手塚宏史, 堀敦史, 石川裕. ワークステーションクラスタ用通信ライブラリ PM の設計と実装. 並列処理シンポジウム JSPP'96. 情報処理学会, June 1996.
3) 手塚宏史, 堀敦史, 石川裕, 曽田哲之, 原田浩, 古田敦, 山田努. PC とギガビット LAN による PC クラスタの構築. 情報処理学会研究会報告 ARC. 情報処理学会, August 1996.
4) Atsushi Hori, Takashi Yokota, Yutaka Ishikawa, Shuichi Sakai, Hiroki Konaka, Munenori Maeda, Takashi Tomokiyo, Jörg Nolte, Hiroshi Matsuoka, Kazuaki Okamoto, and Hideo Hirono. Time Space Sharing Scheduling and Architectural Support. In D. G. Feitelson and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, Vol. 949 of *Lecture Notes in Computer Science*. Springer-Verlag, April 1995.
5) A. Hori, H. Tezuka, Y. Ishikawa, N. Soda, H. Konaka, and M. Maeda. Implementation of Gang-Scheduling on Workstation Cluster. In D. G. Feitelson and L. Rudolph, editors, *IPPS'96 Workshop on Job Scheduling Strategies for Parallel Processing*, Vol. 1162 of *Lecture Notes in Computer Science*, pp. 76–83. Springer-Verlag, April 1996.
6) 堀敦史, 手塚宏史, 石川裕, 曽田哲之, 小中裕喜, 前田宗則. 並列プログラム実行環境のワークステーションクラスタ上での実装. 並列処理シンポジウム JSPP'96. 情報処理学会, June 1996.
7) Hiroshi Tezuka, Atsushi Hori, Yutaka Ishikawa, and Mitsuhisa Sato. PM: A Operating System Coordinated High Performance Communication Library. In *High-Performance Computing and Networking '97*, 1997.
8) Francis B. O'Carroll, Atsushi Hori, Yutaka Ishikawa, and Satoshi Matsuoka. Implementing MPI in a High-Performance, Multithreaded Language MPC++. In *SWoPP'96*, pp. 141–146, 1996.
9) Francis B. O'Carroll, Atsushi Hori, Hiroshi Tezuka, Yutaka Ishikawa, and Satoshi Matsuoka. Performance of MPI on Workstation/PC Clusters using Myrinet. In *Cluster Computing Conference 1997*, 1997.
10) http://lhpca.univ-lyon1.fr/.
11) http://lhpca.univ-lyon1.fr/bip.html.
12) Scott Pakin, Mario Lauria and Andrew Chein. "High Performance Messaging on Workstations: Illinois Fast Messages (FM) for Myrinet". In *Proceedings of Supercomputing '95, San Diego, California*, 1995.