計算機アーキテクチャ 130－25
（1998. 8. 7）

Windows NT クラスタ上での Myrinet による通信の実現

松田元彦† 田中良夫† 久保田和人†
手塚宏史† 石川 裕† 佐藤三久†

RWC PC クラスタ環境は，現在 PC 上の Unix 系 OS 上で実現されている．クラスタ環境の主な構成要素は，Myrinet 通信レイヤ PM，マルチ・ユーザ／マルチ・ジョブ環境 SCore-D，MPI の実装 MPICH-PM である．このうち PM と MPI を Windows NT に移植し簡単な性能測定を行った．移植に際して未解決の課題は Unix のシグナルの代替機能である．性能面では，ユーザ・レベル通信を使用することから OS に依存しないハードウェアの通信性能が得られることを確認した．また OS に依存するページの物理メモリへのロック性能も Unix 系 OS に大差ない結果が得られた．

Communication Library for Myrinet on Windows NT Clusters

Motohiko Matsuda ,† Yoshio Tanaka ,† Kubota Kazuto ,†
Hiroshi Tezuka ,† Yutaka Ishikawa † and Mitsuhisa Sato†

A new port of the RWC PC Cluster Environment on Windows NT is presented. The environment includes a message passing layer PM, a job scheduler SCore-D, and an implementation of MPI MPICH-PM. The unsolved difficulty encountered in the porting is the signal functionality of Unix, which is used to schedule processes in SCore-D. The performance results validate that the bandwidth is bound not by the OS overhead but by the hardware, because PM communication operates in a user-level. The performance results also show that Windows NT performs good enough in locking pagable memory regions.

1. はじめに

PC は急速に高性能化しているが，OS が Windows であることも多い．その Windows も NT になりクラスタを構築するに耐える環境となってきた．そこで現在は Unix 系の OS 上で実現されている RWC PC クラスタ環境を Windows NT へ移植することを試みた．

RWC PC クラスタ環境は[11]，PC 上で一連のソフトウェア群 PM + SCore-D + MPICH-PM によって構成される．これらはクラスタ上でギャング・スケジューリングされるマルチ・ユーザ，マルチ・ジョブを実現し，標準的なメッセージ通信ライブラリである MPI (Message Passing Interface)[1] による高性能な並列プログラミング環境を提供している．

PM はメッセージ・パッシングとリモート・メモリ書き込みによる基本的な通信を提供する．SCore-D はデーモンによるギャング・スケジューリングを行う並列プログラム実行環境である．MPICH-PM はフリーな MPI の実装である MPICH の移植で，通信に PM を使用するようにデータリンク層が変更されている．

† 新情報処理開発機構
Real World Computing Partnership

現在，RWC PC クラスタ環境は Unix 系の NetBSD や Linux などの OS で実現されている．高性能化が進む PC であるが，その OS に Windows が使われていることも多い．この時 OS の入れ替えなしに高性能クラスタ環境を実現できれば望ましい．例えば，オフィスで利用する PC がハイエンドのマルチ・プロセッサ構成であり，OS に Windows NT を使用している場合もある．PC クラスタ環境を Windows NT に移植しておけば，このような PC をクラスタのノードとして取り込むことができる．いくつかの PC クラスタ・プロジェクト (HPVM[2], WinMPI[3], WMPI[4] など) では，Windows NT へのクラスタ環境の移植も進められている．

本稿では Unix 系の OS から Windows NT への RWC PC クラスタ環境の移植した際の問題点と課題およびその基本性能について報告する．

以下，2 節では使用したクラスタの構成と使用したクラスタ・ソフトウェアの概要を述べる．3 節は Windows NT の特徴を紹介する．4 節では移植の詳細について，5 節ではその実行環境を簡単に述べる．6 節では基本性能を示す．7 節では残った課題と対処について 8 節では関連研究に触れる．

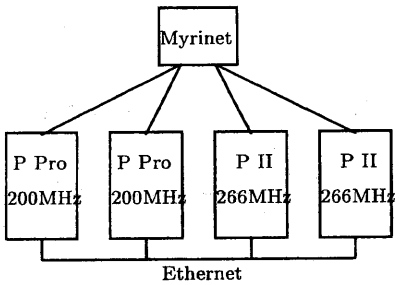

図 1　評価用クラスタの構成

表 1　評価用 PC の仕様

ハードウェア仕様		
CPU	Pentium Pro 200MHz	Pentium II 266MHz
L2- キャッシュ	256KB	512KB
チップセット	450KX	440FX
メモリ	64MB	64MB
バス・バンド幅		
DMA (read)	58.9MB/s	127MB/s
DMA (write)	20.9MB/s	124MB/s
DMA* (read)	15.9MB/s	16.2MB/s
DMA* (write)	10.3MB/s	15.8MB/s
メモリコピー	45.0MB/s	51.2MB/s

* キャッシュ・ヒット時の値

2. クラスタ構成

2.1 システム構成

図 1 に評価用クラスタのシステム構成を示す．後節における性能評価の結果もこの構成を使用した．ノード PC としては現在一般的な Intel 社製 Pentium Pro あるいは Pentium II を使用している．実験にはデスクトップで使用されていた PC を流用した．

表 1 に主なハードウェア仕様を示す．スペック的にハイエンド機種というわけではなく，また変化の速い PC のハードウェアとしてはすでに何世代か前のものになる．表 1 において DMA バス・バンド幅は，データがキャッシュ上にヒット／ミスヒットする両方の場合を示した．キャッシュ・ヒットする場合に性能が悪いことが分かる．コヒーレンス処理が入るので性能が落ちるとも考えられるが，特に Pentium II + 440FX の場合に著しい．ヒット時の測定は，CPU でリードした後に DMA する時間を計測し，リードだけを行った場合の時間を引いてバンド幅に換算している．

通信ハードウェアには Myricom 社製 Myrinet ボード[5] を使用する．Myrinet は通信リンク，ホスト・インタフェース，スイッチから構成される．通信リンクは片方向 160MB/s の速度を持つ 8 bit 並列，双方向のデータパスである．ホスト・インタフェースは PCI バスにより接続される．Myrinet スイッチは 8 × 8 のクロスバで cut-through ルーティングを行う．このスイッチは任意段数を任意のトポロジに接続可能である．ルーティングは，メッセージの先頭に付けられた情報によるソース・ルーティングを使う．今回はスイッチ 1 個使って接続されている．

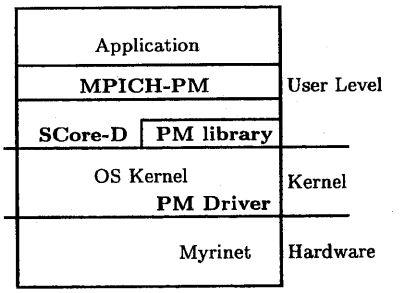

図 2　PM + SCore-D + MPICH-PM ソフトウェア階層

OS には Microsoft 社 Windows NT を使用する．その詳細は，Windows NT 4.0 Workstation に加えマイナーリリースに相当する Service Pack 3, および同社から提供されるユティリティ群 ResourceKit を使用している．さらに OS 以外には，Microsoft 社のプログラム開発環境である VC++, ソフトウェア開発環境 MS SDK, デバイス・ドライバ開発環境 MS DDK を導入している．これらはすべて Windows における標準的な環境といえる．

2.2 クラスタ・ソフトウェア構成

図 2 にクラスタ・システムのソフトウェア構成を示す．それぞれのレイヤは以下の機能を持つ．

PM 通信レイヤ. PM は Myrinet 上の通信レイヤであり，低レイテンシなメッセージ通信と高バンド幅なリモート・メモリ書き込みを提供する[6]．PM は Myrinet NIC (Network Interface Card) 上のマイクロ・プロセッサ用のプログラムとライブラリから構成される．通信はユーザ・レベルから直接 NIC をコントロールすることによって処理し，システム・コールや割り込みといったオーバヘッドをなくしている．また PM では，リモート・メモリ書き込み時に必要となるページの物理メモリへのロック (ピンダウン処理) のコストを低減するため，ピンダウンされる領域を LRU (Least Recently Used) で管理している[6]．

SCore-D. SCore-D は RWC PC クラスタ環境の中心に位置するスケジューラを含む実行環境である．ユーザ・レベルのデーモンを使用して，PC クラスタでのマルチ・ユーザ，マルチ・ジョブ環境を実現している．ユーザの並列ジョブは時間／空間分割され，それぞれギャング・スケジューリングされる[7]．プリエンプション可能なネットワーク・コンテキスト保持機能により，効率良いプロセス切り替えを実現している[8]．この他，プロセスの負荷等のダイナ

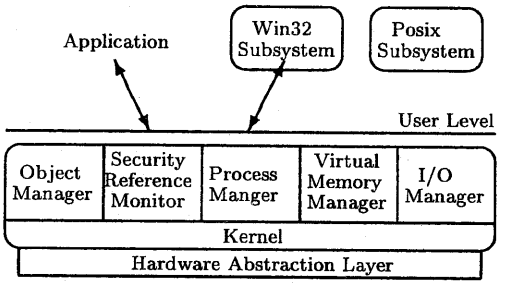

図 3 Windows NT のシステム階層

ミックな実行状況のモニタ機能も持っている.

MPICH-PM. MPI[1] の実装は, Argonne National Laboratory と Mississippi State 大の開発した MPICH[9] をベースにしている. MPICH は ADI (Abstract Device Interface) という中間的なレイヤを定義することで移植性を高めている. つまり ADI のレイヤを実装すれば, MPI の実装が完了するようになっている. MPICH-PM では, その ADI を PM によって実装している. 特に ADI の Channel Interface を使っている. ADI における Eager 通信は PM のメッセージ通信で, Rendezvous 通信は PM のリモート・メモリ書き込みで実現されている[10].

ここで使用したベースとなる PM, SCore-D, MPICH-PM について, これらソフトウェアは新情報処理開発機構から無償で入手可能である[11]. 今回はこの配布版をベースに移植を行った.

3. Windows NT

3.1 Windows NT の概要

Windows NT はマルチ・プロセス, マルチ・ユーザの OS で, 特にセキュリティ (C2 レベル), ネットワーキング, マルチプロセッサ (SMP) に対応している. プロセス間の関係やファイル・システムに関しては複数のモデルを提供している. これらはサブシステムと呼ばれ, Posix[12] や Windows といったモデルが提供されている. このような複数環境は, 標準仕様である Posix への要請と Windows とのコンパチビリティの要請から提供されている. Windows とのコンパチビリティに関してはマーケッティングに問題のあった OS/2 の反省から来ている.

Windows NT 内部はマルチスレッドによるマイクロ・カーネルと, オブジェクトによる資源管理を行うモダンな OS である[13]. ただしファイル・システムやデバイスなどはカーネル内に置かれる. さらに性能上の問題からディスプレイの表示処理やウィンドウ・マネージャがカーネル内に置かれている. 一方, プロセスやファイル・システムのモデルに関しては, カーネル外に置かれるサーバ・プロセスによってその意味が与えられる. Windows NT ではカーネルへの直接のインタフェースは公開されていない. システム・コールはサブシステムとともに提供される API によって行う. 図 3 に参考文献[13] による Windows NT のシステム概要を示す.

3.2 システム構成

今回の実験環境には基本的な環境, Microsoft 社の提供する環境だけを使用した. Windows NT では複数のプログラム実行環境をサブシステムという名称で提供している. 代表的なものは下にあげる 3 システムがある. 今回はこの中で Win32 という Windows NT のネイティブ環境を使用した. 以下にそれぞれの特徴をあげる.

Win32 環境. Windows NT のネイティブ環境であり, 他の Windows システムとコンパチブルである. 当然この環境からは Windows NT の全機能が利用できるとともに, Microsoft 社の提供するソフトウェア開発環境などもこの環境を対象としており充実している.

Microsoft Posix 環境. Windows NT 上ではインストール時点で Posix 環境が提供されている. これは完全な Posix 準拠の環境で, Posix 準拠ライブラリとサブシステムを実現するためのサーバより構成される. この Posix 環境は完全な IEEE Std POSIX 1003.1-19880[12] 準拠な環境を提供している. Posix プログラムはバイナリ・ファイルに付けられたマークにより起動時に Posix サブシステムの元に実行されることになっている.

ただし, 各サブシステムは独立しており排他的にサーバにアクセスするので, Posix 環境から Win32 で提供されている機能を直接利用するようなことは不可能になっている. 唯一可能なのは標準入出力 (stdio) を使った「パイプ」によるデータ交換だけである.

OpenNT 環境. OpenNT はサードベンダから提供される商用のサブシステムで, BSD 系の Unix 環境を提供する. これは Microsoft 社の Posix サーバを独自のサーバで置き換えること, コマンドやライブラリ群を提供することにより行われる. 標準的なデバイス (テープ, pty 等) は提供されるが, Windows NT のデバイスは直接サポートされない. また, サードベンダの商品であり標準的とはいえない.

PM + SCore-D + MPICH-PM は Unix 系の OS 上で開発されているのでまず Posix サブシステムを使用することが考えられる. しかし Microsoft の提供する環境は純粋な Posix 環境のみであるので目的にそぐわない. Posix では I/O としてファイル・システムとコンソールのみを定義するだけである. すなわち, デバイスへのアクセスを提供しないので一般プログラムの利用には使用できない.

4. 移植の詳細

移植に際して最も目だった Unix 系と Windows 系の OS 機能の差異は, Unix における非同期イベント通知であるシグナル機能であった. Windows NT ではイベントは基本的にスレッドによる待ち受けかポーリングを使用するので, シグナルに相当する機能がない.

以下に移植の詳細をソフトウェアのレイヤに従って示す.

4.1 デバイス・ドライバ

デバイス・ドライバは NetBSD 版のものを手本にほぼ 1 対 1 に対応する形で実装されている. Windows NT ではデバイス・ドライバへのインタフェースとして提供される機能は豊富であり, かなり高機能なことも可能である.

PM のデバイス・ドライバとして特に要求される機能としては, 以下のものがある:

- DMA に使用する, 連続する物理アドレスを持つメモリ領域のアロケーション
- ユーザ・プロセス中の自由なアドレス領域の物理アドレスへのロック (ピンダウン処理)

これらは, それぞれ `MmAllocateContiguousMemory` と `MmProbeAndLockMemory` という機能で提供されており問題ない. また, I/O デバイスに対するユーザ・プロセスのアドレス空間へのマッピング (`mmap/munmap`) という概念は Win32 自体にはないが, カーネル内のドライバ・インタフェースとしては `ZwMapViewOfSection` いう機能で提供されている. これは任意のユーザ・プロセス間でアドレス空間のマッピングが可能である. また物理アドレスを使ったマッピングの指定も可能である.

残る未実装の機能としては Unix でいうところの `select` と, 非同期イベント通知であるシグナルがある. Windows ではセマフォを使うので `select` はない. またシグナルも Windows にはない概念である. 現在のところ, ユーザ・プロセスへのシグナルを使った通知機能は実験用に使われているだけで, SCore-D や MPICH-PM では使用されていない.

4.2 PM 通信ライブラリ

Myrinet NIC 上のプロセッサのコードに変更はない. 変更なしでバイナリをそのまま使用している.

ライブラリ・コードの変更は概ね単純な置き換えだけであった. インクルード・ファイルの変更, ごく少数の関数名や型名についての名称変更, そして若干の引数の違いの変更である. 例えば, 変更が必要であった関数名には以下のものがある.

Unix	Windows NT
`open`	`CreateFile`
`close`	`CloseHandle`
`ioctl`	`DeviceIoControl`

当然であるが, C コンパイラと標準ライブラリは ANSI C に準拠しているので基本的な部分について変更は全く必要でない.

また, Win32 にはデバイスのマッピング (`mmap/munmap`) の概念がないので, これは `DeviceIoControl` によりインタフェースをとった.

4.3 SCore-D スケジューラ

SCore-D では, ジョブ・スケージューリングを Unix のシグナルを使ってユーザ・レベルで実現している. 具体的には, STOP と CONT (-inue) の 2 つのシグナルを使って実行可能なプロセスを切り替えている[7]. このように SCore-D は Windows に欠けている機能を使用しており代替実現方法を必要とするので, 今回の移植では扱わず課題として残った.

SCore-D は RWC PC クラスタ環境で中心的な位置を占めるが, 他の通信ライブラリ部分とはほぼ独立に実現されているので, 通信機能のみの移植だけでも動作可能であった.

4.4 MPICH-PM

MPICH 本体は ADI へのコールだけで閉じているので, MPICH-PM の配布版から変更の必要は全くなかった. ただし Windows のコンパイル環境である VC++ では, Unix で一般的に使われる make とは異なる nmake というものを使用する. そのため `Makefile` の書き直しが必要であった.

PM を使って記述された ADI 部分に関しても, 基本的には配布版からの変更は必要なかった. ただし, 今回は SCore-D の移植を先送りにしたのでそのため若干の変更が必要であった. 変更点にはスケジューラに対してヒントを教える `_scored_become_idle` および `_scored_become_busy` が含まれる. これらについてはダミー・ルーチンを用意した.

5. 実行環境

MPIRUN 等の機能のために, プロセスをリモートで起動する必要があるが, リモート・シェル (`rsh`) などは Microsoft 社から提供される. これらは「ResourceKit」という名称の無償で提供されるユーティリティ群に含まれている. ResourceKit にはリモート・シェル以外に, リモート・コピー (`rcp`), リモート・コンソール (`rclient`) など必要となる最低限の機能が含まれている. 今回はこれらのコマンドを基本環境をみなした.

ただし, ResourceKit は試験的に提供されている面があり, 例えば `rsh` は約 10 分で接続が解除される仕様になっている. 実験程度なら ResourceKit で提供される機能で十分であったが, 実用には必要なコマンド群をそれぞれ用意する必要があろう.

6. 基本性能

PM はユーザ・レベル通信であるので OS に依存し

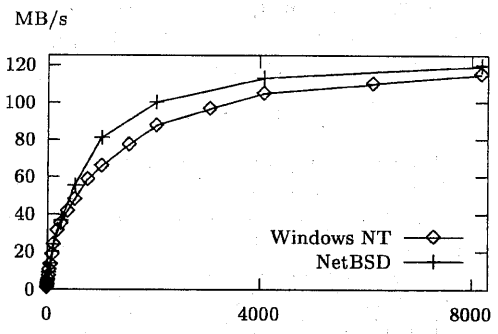

図 4 PM におけるバンド幅の比較．グラフに現れる性能差は，測定に DMA のピーク性能が若干異なるハードウェアを使用したことに起因する．

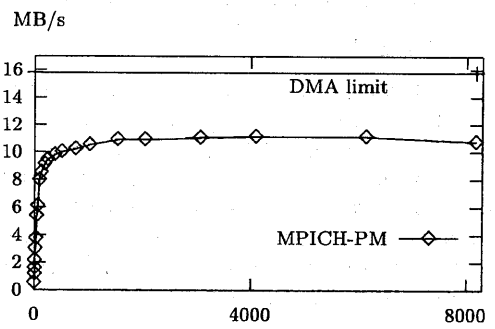

図 5 MPI におけるバンド幅．グラフには DMA ハードウェアの性能限界を示した．

表 2 ページ・ロックの性能比較．

操作	Windows NT	Linux
mlock (μsec)	58.6 (1.4)	40.6 (1.0)
munlock (μsec)	48.5 (1.1)	45.0 (1.0)

ページあたりのピンダウン処理時間．処理時間には LRU 処理も含む．計測は Pentium Pro (200MHz) 上で行った．カッコ内は Linux との比．

ないハードウェアの性能がそのまま観測されるはずである．つまり OS が関与するのは初期化部分だけであり，通信処理部分では直接ハードウェアを操作しているので OS の違いによる影響はない．以下ではこのことを実測により確認する．

ただし，ページの物理メモリへのロック (ピンダウン) 操作は通信処理中に呼ばれる可能性があり，極端に遅い場合には性能に影響を与える可能性がある．これについては操作にかかる時間を個別に計測し，他の Unix 系の OS との時間を比較する．

6.1 バンド幅

図 4 に PM を直接使用した通信バッファ間でのデータ転送バンド幅を示す．Pentium II (266MHz) のノード間で測定した．

PM では物理メモリにロックされたメモリ領域を通信バッファとして使用する．ユーザ・プロセスはこの領域をマッピングして使用する．ここでの測定は，このバッファ間の転送バンド幅を示している．

結果は PCI の DMA 転送速度に制限されている．Myrinet のリンク速度 (160MB/s) は PCI の速度 (133MB/s) より十分速い．比較には NetBSD での結果を用いた[14]．Windows NT も NetBSD も PCI のピーク性能に近い値が出ている．ただし，グラフに現れている差は Windows NT と NetBSD で若干 DMA のピーク性能が異なるハードウェアで測定しているためである．

6.2 ページ・ロック

表 2 にページ・ロックの性能比較のため，Pentium Pro (200MHz) 上で Windows NT と Linux による測定結果を示す．具体的にはピンダウン・オペレーションの時間は，ライブラリでの LRU によるマネージメント時間とシステム・コールの時間を加えたものになる．表では，1024 ページをページ単位でロックする時間を計測したものを 1 ページあたりの時間に換算している．単位は μsec である．各ページはあらかじめアクセスしてロードされている状態にしてから計測した．

Windows NT は Linux に比べて 1.4 倍程のコストがかかっている．しかし MPICH-PM の実装ではピンダウンされるエリアは LRU で管理されており，実際のアプリケーションでは LRU のヒット率が高いことが示されている[14]．この程度の差は実際のアプリケーションでは許容範囲だと考えられる．

6.3 MPI の性能

図 5 に MPICH-PM を使用した場合の Point-to-Point バンド幅を示す．Pentium II (266MHz) のノード間で測定した．これは一方のノードが `MPI_Send` を繰り返し，他方のノードは `MPI_Recv` を繰り返すだけのプログラムで測定した．送信側も受信側もそれぞれ，一回の転送につきコピーと DMA 転送を一回ずつ行う．コピーは PM のバッファ領域へのデータ転送のために必要である．

ここでの値は文献 10) で報告されている値 (約 50MB/s) に比べ極端に性能が悪い．これは前述のように実験に使用したハードウェアの DMA 性能が悪いことに起因する．本来ならばメモリ・バンド幅がネックになるが，データをバッファ領域へコピーするためデータがキャッシュに載り，DMA 性能が落ちていると考えられる．

6.4 性能比較

予想通りハードウェア性能に近い基本性能を実現している．ただし使用した PC のハードウェアの性能が思ったより悪かったため Unix 系の OS と比較ができず，評価は十分とは言えない．ページ・ロックに関しては Unix 系の OS に比較して実際のアプリケーション

では問題ない程度の性能が出ていることが確認できた.

7. 課題と対処

残った課題としては, SCore-D のスケジューラがある. SCore-D のスケジューリングは Unix の非同期シグナルを使ったプロセス管理を行っているが, Windows にはこの機能がない. かわりに Windows のプロセス管理が, プライオリティ・ベースであることを利用して実現することが考えられる. すなわち, デスケジュールされたプロセスのプライオリティを大きく下げることで所望の効果を得ることが可能である[15].

8. 関連研究

MPICH/NT (WinMPICH)[3]. MPICH の Windows NT への移植である. MPICH の開発者の Mississippi State 大で開発された. 元々は SMP での実行が提供された. 現在開発は終了しており, 古い MPICH をベースにしたものが配布されている. これをベースに MPI/PRO[16] として商用化されている.

WMPI[4]. これも MPICH をベースにしたもので, ポルトガルの Coimbra 大で開発された. これは ADI に p4 デバイスを用い, TCP/IP により通信を行う.

HPVM[2]. Myrinet を使用する Windows NT クラスタであり, これも MPICH を移植したものを提供する. 通信には FM (Fast Message) というポータブルな層を使っている.

これらの Windows NT 上の MPI の性能比較が文献 17) にある.

9. おわりに

RWC PC クラスタ環境の Windows NT への移植を行い, 簡単な性能を示した. 基本的にユーザ・レベル通信なので OS が関与するのは初期化等の部分のみで, 性能に影響を与える部分は皆無であるので性能は予想通りであった.

Windows NT を使用する上で本質的な問題点には出会わなかった. Windows NT はマルチユーザの OS として設計されており, プログラミング・モデルと環境の違いを除くと, 機能的には Unix 系の OS と遜色なく使用できた.

一番問題になったのは Windows でプログラミングの経験がないこと, さらに近辺に Windows でのプログラミング経験者が皆無であったことである. しかし総じてみると, 当初懸念したのに比べ全くと言って良いほどトラブルもなく移植が進んだ. ただし全体的には, ドライバ開発からライブラリ, 実行環境へと開発が進むにしたがい, 標準的に提供されるユーティリティ群が非力になり困難や煩雑さを伴ってくる印象を受けた.

今後は, SCore-D の移植, およびコンソールを使っているユーザのジョブ等のスケジューリングを含む環境として完成させることが課題である.

参考文献

1) Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard, May 5, 1994.* University of Tennessee, Knoxville, Report CS-94-230, 1994.
2) HPVM. `http://www-csag.cs.uiuc.edu/projects/clusters.html`.
3) MPICH/NT (WinMPI). `http://www.erc.msstate.edu/mpi/mpiNT.html`.
4) WMPI. `http://dsg.dei.uc.pt/w32mpi`.
5) `http://www.myri.com`.
6) 手塚, 堀, 石川. 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, pp.41–48, 情報処理学会, June 1996.
7) 堀, 手塚, 石川, 曽田, 小中, 前田. 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, pp.49–56, 情報処理学会, June 1996.
8) 堀, 手塚, 石川. ギャングスケジューリングの高速化技法の提案. 並列処理シンポジウム JSPP'98, pp207–214, 情報処理学会, June 1998.
9) MPICH. `http://www-c.mcs.anl.gov/mpi`.
10) F. O'Carroll, H. Tezuka, A. Hori, and Y. Ishikawa. The Design and Implementation of Zero Copy MPI Using Commodity Hardware with a High Performance Network. In *Intl. Conference on Supercomputing '98*, July 1998.
11) Real World Computing Partnership. `http://www.rwcp.or.jp/lab/pdslab`.
12) IEEE Std 1003.1-1990. *Standard for Information Technology — Portable Operating System Interface (POSIX).*
13) H. Custer. *Inside Windows NT.* Microsoft Press, 1993.
14) 手塚, 堀, F. O'Carroll, 石川. RWC PC Cluster II の構築と性能評価. 情報処理学会研究会 HOKKE'98, March 1998.
15) M. Buchanan, and A. Chien. Coordinated Thread Scheduling for Workstation Clusters Under Windows NT. In *Proc. USENIX Windows NT Workshop*, 1997.
16) MPI SoftwareTechnology, Inc. MPI/PRO for Windows NT. 製品説明. `http://www.mpi-softtech.com`.
17) M. Baker and G. Fox. MPI on NT: A Preliminary Evaluation of the Available Environments. IPPS Workshop PC-NOW'98, LNCS 1388, Springer-Verlag, March 1998.