並列計算機システムFOLON上へのPVMの移植と評価

菊地 賢太郎† 外山 孝伸† 大原 拓三††
土山 了士† 小林 一成†
上田 学† 上田 和紀††

並列計算機システムFOLONは既存のハードウェアを利用して構築した共有メモリを持つシステムである。共有バスにVMEバスを使用しPEにPCを使用することで、ハードウェア工作なしで、レイテンシが4.2μsの通信を実現している。FOLON上に基本的なメッセージパッシングモデルの通信ライブラリFCPを構築し、それを用いてより一般的なPVMの並列実行環境及びライブラリを実装している。現在FOLONにおいては、排他処理をソフトウェアとして、最下位ライブラリである実現しており、FCPのレイテンシが200μs程度になっているが、排他処理の最適化によって今後かなりの性能向上が見込まれる。本稿ではこれら通信ライブラリの実装の現状と今後について述べる。

Design and evaluation of PVM on the parallel computer system FOLON

KENTARO KIKUCHI,† TAKANOBU TOYAMA,† TAKUZO OHARA,††
RYOJI TSUCHIYAMA,† KAZUNARI KOBAYASHI,† GAKU UEDA†
and KAZUNORI UEDA††

FOLON is a shared-memory parallel computer system consisting of high performance PCs and VMEbus. Although built without developing dedicated hardware, FOLON achieves the communication latency of 4.2 μs. A basic interprocess communication library, FCP, is provided to use FOLON-specific functionalities. We have implemented PVM using FCP. Currently, mutual exclusion management is provided by software in VCL, the lowest-level library of FOLON, and the communication latency of FCP is about 200μs. However, we estimate that latency of FCP could be considerably shortened by optimizing the exclusion management. This paper describes the design of those libraries and their performance.

1. はじめに

最近はパーソナルコンピュータ(以後PCと略)の性能が劇的に上がり、非常に性能の良いコンピュータが安価に入手できるようになっている。我々は1994年末から、これらPCと既存の通信ハードウェアを組み合わせて並列計算機システムFOLONを構築している。FOLONは以下のような設計方針で開発している。

- PCをPEとすることで、システム全体を安価に構築すると共に、PCの性能向上に追随したアップグレードを可能にする
- 新規のハードウェア製作を行わずに構築する
- 並列計算機システムとして必要だが、ハードウェアで持たない機能についてはソフトウェアで実装する
- 細粒度並列処理に適したプラットホームを目指し、Ethernetによるワークステーションクラスタよりも高い通信性能、特にレイテンシの大幅な向上を図る

この方針に従い、PEとしてPCを使用し、共有バスとしてVMEバスを使用したシステムを構築することで、Ethernetを使用したワークステーションクラスタに比べて、非常にレイテンシの小さい通信を実現した[4]。

本研究では、このFOLON上にメッセージパッシングプロトコルFCPおよびインターフェースとしてPVM[3]を実装する。FOLONの性能特性を活かすという観点からは、PVMの利用は必ずしも最適であるとは限らないが、

- 既存のPVMを使ったアプリケーションを容易に移植し、また効率よく実行する
- FOLONの特性を活かした並列化ライブラリの設計と実装の出発点とする

† 早稲田大学大学院理工学研究科
Department of Information and Computer Science, Graduate School of Science and Engineering, WASEDA University
†† 早稲田大学理工学部情報学科
Department of Information and Computer Science, School of Science and Engineering, WASEDA University

という観点から、まずPVMのようなライブラリを実装することは重要である。本稿では、メッセージパッシングモデルライブラリFCPおよびPVMの実装技法について述べ、各レベルでの通信性能について述べる。

ワークステーションやPCを構成要素として、ハードウェア工作を行なわずに(あるいは最小限にとどめて)高通信性能の並列計算機を構成する技術は、並列計算機の普及のために非常に重要になってきている。本研究で採用した接続ハードウェア(第2節)は、並列計算機構築用として提供されているものではないが、その後、Myrinetなど、並列計算機の構成に適したいくつかの相互接続ハードウェアが出てきている[5]。VMEバスを用いたワークステーションクラスタの研究としては[2]がある。[2]では相互排他のためのハードウェアを追加しているが、我々は、ハードウェア工作なしでどの程度の性能が達成可能かを追求している。

2. 並列計算機システムFOLONの概要

2.1 FOLONのハードウェア構成

FOLONはPEとしてIBM PC/AT互換のパーソナルコンピュータ(PC)を使用している。現在図1のように、6台のPentium120搭載のPCで構成されている。

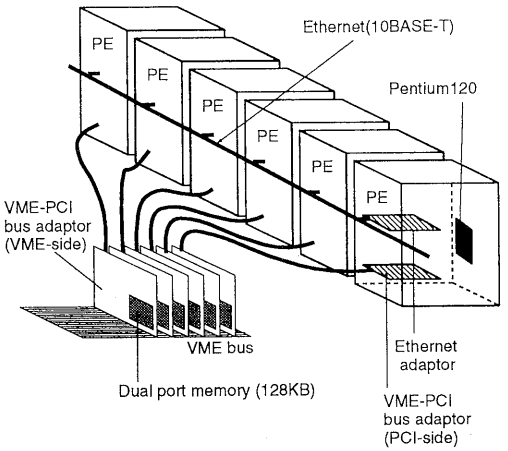

図1 FOLONのハードウェア構成

FOLONでは、外部バスとしてVMEバスを採用しており、VMEとPCはBiT3社製のVME-PCIバス変換アダプタModel 617を用いて接続されている[1]。バス変換アダプタ上には、デュアルポートメモリを各128KBずつ搭載している。このデュアルポートメモリのアクセスタイムの公称値は、自PEのメモリに対しては400ns、他PEのメモリに対しては2.1μsである。このメモリは各PEのメモリアドレス空間とは独立した、VMEメモリ空間上に配置され、バス変換アダプタを通して各PE間で共有させている。大量のデータ転送にはDMA転送を用いることで、高速な処理を行うようにしている。

2.2 FOLONのソフトウェア構成

FOLONの各PEにはOSとしてFreeBSDを使用している。並列計算機システムとしての機能はOS上に実装されたデバイスドライバと複数のライブラリ類で実現する。またFOLONで使用しているハードウェアは、並列計算機として動作させるために開発されたものではないので、Test & Setなどハードウェア的に実装していない機能は、ソフトウェアで実現することになる。現在のFOLONのライブラリ構成は図2のようになっている。

デバイスドライバ VMEバスおよびバス操作のためのハードウェアの制御

VCL (VME Control Library) 共有メモリへのアクセスおよびVME資源の制御

FCP (FOLON Control Protocol) メッセージパッシングモデルのプロトコルおよびライブラリ

2.2.1 デバイスドライバ

デバイスドライバはVMEバスアダプタのコントロール、共有メモリのマッピング、DMA転送、VMEバスからの割り込み処理を担当している。

メモリマッピングは各PEのバス変換アダプタ上に搭載されたデュアルポートメモリを各PEのメモリ空間の空き領域に共有メモリとしてマップする。このメモリをOSをバイパスしてアクセスすることで、高速な通信が可能になる。

またVMEバスアダプタはDMA転送機能を持っている。これを使うことで、大量のデータ転送の場合のスループットを上げることができる。ただしDMA転送は、転送データの含まれる物理メモリ空間のスワップアウトを禁止する必要があるため、デバイスドライバとして実装している。

2.2.2 VCL(VME Control Library)

VCLはFOLONの持つ共有メモリへのアクセスをOSをバイパスさせてユーザープロセスに提供するライブラリである。

またこの共有メモリへのTest & Setを提供している。Test & Setは、OSを用いてPE内のプロセス間で排他を行い、さらにVMEのバスロック機能を用いて全体の排他を行なうことで実現している。

3. メッセージパッシング型通信ライブラリFCP

3.1 メッセージパッシングモデルの必要性

VCLによって、共有メモリへのアクセスや基本的な処理のための手段は用意されている。だが、PVMやMPI[6]などのメッセージパッシングモデルの並列ライブラリを移植することを考えた場合、メッセージパッシング型の中間ライブラリを実装することによって、比較

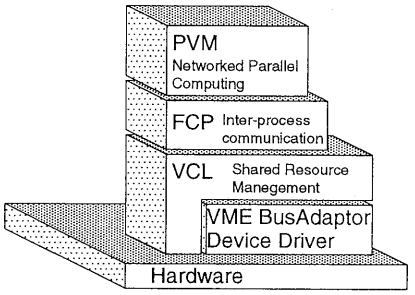

図2 FOLON のライブラリ階層

的容易にこれらのライブラリを移植できるようになる。なお、メッセージパッシングモデルと共有メモリモデルは共存できる。FOLON では、共有メモリの一部をメッセージパッシングモデルにおけるバッファとして使用し、他の部分は共有メモリとして使用するように提供することができる。

以上のような理由から、メッセージパッシングモデルのプロトコル FCP(FOLON Control Protocol) を実装した。

3.2 FCP が提供する機能

我々はメッセージパッシングモデルの中間ライブラリとして、FCP(Folon Control Protocol) を設計し、これにより複数のプロセスがそれぞれ持つ受信バッファを管理することにした。

FCP では、主に

- 通信用のバッファを獲得 / 開放する `FCP_alloc()`, `FCP_free()`
- 相手のバッファにデータを書き込む `FCP_send()`
- 自分のバッファに届いているメッセージを読み出す `FCP_recv()`
- その他、動作中のプロセスに関する情報を取得するなどの FOLON に特化した機能

などの関数・機能が用意されている。

3.3 FCP の設計方針

FCP は、以下のような方針で設計されている。

- マルチユーザに対応
- 各プロセスは受信バッファのみを持ち、送信バッファを持たない
- データの到着保証および、データの到着順序保証
- 複数プロセスが同時に共有メモリを使用して、安全に通信が実行できるようにする
- UDP, TCP と同等の通信機能を実装、提供

通信プロトコル	FCP	UDP	TCP
コネクションの確立	不要	不要	必要
データの到着保証	○	×	○
データの到着順序保証	○	×	○

表1 FCP と UDP および TCP の比較

3.3.1 バスの混雑の緩和

FCP での通信において、受信プロセスのみが自 PE のバス変換アダプタ上に実装されたデュアルポートメモリに受信バッファを確保する。自 PE のデュアルポートメモリへは共有バスを経由せずアクセスできるので、バス上には送信側のアクセスのみが現れることになる。

3.4 FCP によるプロセスの管理

FCP では、プロセスの受信バッファの情報を管理するテーブルを、共有メモリ上に作成する (以後バッファテーブルと呼ぶ)。このバッファテーブルでは以下のような情報を管理している。

- 各プロセスの受信バッファがどこにあるか (メモリアドレスや CPUID)
- FCP を使用する際に割り当てられる ID(FOLON プロセス ID)
- 複数プロセスが同時に FCP を使用する際に、自プロセスの属するグループを示す ID(グループ ID)

送信時にこのバッファを参照することで、データの到着保証等を行っている。

3.5 改良 FCP, FCP plus について

現 FCP の設計は拡張性、汎用性を求めたため、以下のような点がオーバーヘッドの原因となっている。

- バッファテーブルのような管理構造を共有メモリ上において、各バッファの情報を参照してから通信していること
- バッファテーブル、受信バッファなどのプロトコルとしての管理情報が大きいこと

以上のような理由により、満足できる速度が達成することが困難であると予想されるため、当初の FCP の目的とバッファテーブルの設計を以下のように変更した FCP plus を実装した。

- FOLON システムで並列プログラムを実行するのは、一度に 1 人のユーザだけとする。
- バッファテーブルは生成するが、必ず 1 人しかユーザがいないことを保証して、バッファテーブルの情報を、キャッシュしておく。

PVM の開発で、すでに FCP を使用して動作している (複数のユーザがいることを前提とした) 関数があることを考慮して、FCP の関数 と FCP plus の関数は同時に使用しても共存できるよう設計してある。そのため、プロトコルで決められた管理情報の圧縮は FCP plus では行っていない。FCP と FCP plus で発生する転送量の違いについては後述する。

4. FOLON 上への PVM の移植

PVM(Parallel Vartual Machine) は、Ethernet で接続されたワークステーションクラスタを、単一の仮想並列計算機として利用することを可能とするソフトウェアシステムである。PVM を FOLON 上へ実装することにより、FOLON を汎用的な並列計算機環境として

も動作させることができる。

4.1 PVMの構成

PVMソフトウェアシステムは、デーモンとユーザライブラリの2つから構成される。

4.1.1 Pvmd (PVM Daemon)

PVMデーモン（以後、Pvmdと呼ぶ）は、仮想マシンを構成するすべてのコンピュータ上に常駐し、タスクの管理、タスクからのメッセージの処理、ルーティング、エラー処理等を行う。

通常Pvmdは`select()`システムコールによってポーリングし、タスクや他のデーモンからのメッセージの到着を待つ。メッセージが届くと、メッセージの種類に応じて新たなルーチンを呼び出す。処理が終わると、再びポーリングに戻りこの動作を繰り返す。

4.1.2 ユーザライブラリ

ユーザライブラリは、メッセージやデータの送受信、プロセスの生成、管理等に必要なライブラリ群を提供する。

ユーザはこのライブラリをアプリケーションプログラムにリンクすることにより、仮想並列マシンとしての機能を利用することができる。

4.1.3 PVMでの通信

Pvmd-Pvmd間はUDPソケットを用いて通信される。またPvmd-Task間の通信はTCPソケットを通じて行われる。Task-Task間の通信は標準では、Pvmdを介して行われており、直接は通信していない(図3)。

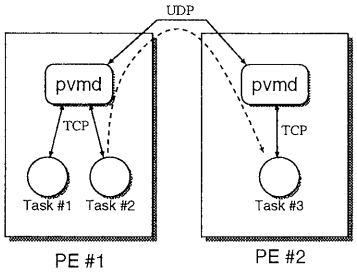

図3 UDP/TCPを用いたTask間通信

4.2 実装の方針☆

PVMでは通信にEthernetを用いているが、これを並列計算機の通信路と考えると、OSを介して通信を行うのでオーバヘッドがかなり大きい。FOLON上に実装されたメッセージパッシングプロトコルFCPを用いることにより、通信のオーバヘッドが減り、PVMのパフォーマンスの向上が見込まれる。FCPライブラリは、UDPプロトコルによるソケット通信との互換性を考慮して開発されている。したがって、PVMのソケット通信の部分を、FCPライブラリによる通信に変更することは比較的容易である。

PVMでおこなっている通信を全てFCPを用いて行うことをめざすが、現在は第一段階としてUDPを用いているPvmd-Pvmd間の通信をFCPによる通信に変更している。Pvmd-Task間の通信は、`socket`による通信をそのまま用いている(図4)。

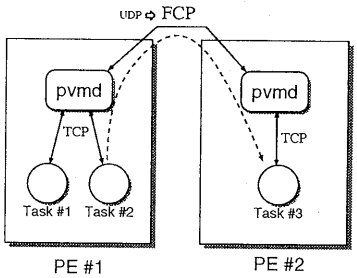

図4 FCP/TCPを用いたTask間通信

4.3 実装法の詳細

通信バッファの確保 FCPは各プロセスごとに受信バッファを持ち、送信側が受信側のバッファに書き込むことで通信を実現する。したがって各daemonが起動時に共有メモリ空間上にバッファを確保するようにする。(`FCP_AllocateBuffer()`)

送信 PVMではデータをパックしてから送信する. FCPの送信関数(`FCP_Send()`)は任意サイズのデータを送信することができるように設計している。そのためPVMのパックされたデータをそのまま送信することが可能になっている。

受信 受信は、自分のバッファに届いているデータを読んでくるだけで完了する。実際にはFCPの受信関数`FCP_Receive()`を用いている。

受信バッファのポーリング PVMでは、データの到着を調べるために`select()`が用いられている。そこでFCPにも到着メッセージを調べるための関数`FCP_Select()`を用意した。この関数を使うことで、自バッファへ送信されたデータの有無を調べることができる。

5. 性能評価

5.1 FOLONの基本通信性能

FOLONシステムにおいて基本ライブラリとなるVCL及びFCPを使用し、通信性能の測定を行った。レイテンシは4バイトのデータを2台のPE間で往復させたときに要する時間から算出し、スループットは4096バイトの転送を行うのに必要な時間から算出した。DMAを用いた転送は、システム全体の安定性に問題があるため今回は実装を見送っている。また、

☆ 移植するソースは、現時点（平成8年7月）で最新であるPVM3.3.11を用いた。

VMEバスを経由した通信と比較するため、Ethernetを経由するUDPを使った通信性能も測定した(表2)。

通信手段	レイテンシ (μs)	スループット MB/s
VCL	4.2	1.7
FCP	240	1.5
FCP plus	200	1.5
Ethernet	320	0.76

表2 各レイヤにおける通信性能

レイテンシ性能は、UDPによる転送と比べ、VCLレベルの転送で76倍、FCPで1.3倍、FCP plusで1.6倍それぞれ向上していることがわかる。FCPレベルの転送が、VCLのみを用いた転送に比べてレイテンシ性能が低下するのは、1回のデータ転送毎に次にあげるようなオーバヘッドが存在するからである。

- VMEバス上の共有資源に対する排他制御
- 受信バッファを制御するために用いられるヘッダの転送
- データを転送する受信バッファの検索

スループットに関しては、VCL/FCPレベルどちらの転送においても、UDPによる転送と比べほぼ2倍になっている。FCPレベルにおいてもVCLのみを用いた場合とほぼ変わらないスループットが得られている。これは、FCPを用いて転送を行う場合、転送するデータの大きさに関わらず、1回の転送にかかるオーバヘッドが一定であるという理由による。今後はDMA転送を使用することにより、さらにスループット性能を向上させる予定である。

5.2 PVMにおける性能評価

Ethernetを経由したUDPを用いた場合と、VMEバスを経由したFCPを用いた場合それぞれの、現段階でのPVMの通信性能を測定し比較した。レイテンシは4バイトのデータを2台のPE間で往復させた時間から算出し、スループットは4096バイトの転送を行うのに必要な時間から算出した(表3)。ただし、FCP plusを用いたPVMの実装は行われていないので、ここではFCPを用いた値である。

通信手段	レイテンシ (μs)	スループット MB/s
Ethernet(UDP)	580	0.60
VME(FCP)	630	1.0

表3 PVMの通信性能

FCPを用いた転送の場合、レイテンシ性能に関してはUDPを用いた転送と比べて1.1倍性能が低下していることがわかる。この原因と対策については考察で述べる。

スループットに関しては、UDPを用いた転送と比べてFCPを用いた転送が1.7倍になっており、基本通信性能の測定において得られた約2倍のスループット性能の向上と近い結果を得ることができた。

6. 考　　察

6.1 基本ライブラリの高速化

VCL/FCPレベルの転送において、レイテンシ性能を下げている主な原因は次にあげられる3点である。

- 排他制御にかかるオーバヘッドが大きい(VCL)
- 送信の際にバッファテーブルを必ず参照するため、バッファテーブルへアクセスが集中している(FCP)
- 各受信バッファを管理するためのヘッダが長い(FCP, FCP plus)

6.1.1 排他制御について

排他制御に関しては、同一PE内のプロセス排他を、カーネル内に置いた排他フラグをシステムコールでTest & Setすることによって行っており、排他制御のオーバヘッドが大きくなる原因になっている。

同一PE内のプロセスで共有できるメモリ上にフラグをおき、メモリアクセスのみのTest & Setを実装することで、排他制御にかかるオーバヘッドを約半分に減らすことができると見込んでいる。

6.1.2 プロトコルの最適化

これまでの問題点をふまえて、FCPの最適化を行っている。最適化されたプロトコルをFCP IIと呼び、以下のような方針で設計している。

- 受信バッファの静的配置 / 上位ライブラリに依存した、バッファテーブル管理の削減
- 受信バッファの管理領域の冗長な情報の圧縮による高速化

6.1.2.1 受信バッファの静的配置について

FCP IIでは上位ライブラリの特性に依存させて、バッファテーブルを用いた管理を削除、もしくは最小限にすることを考えている。

PVMを実装する場合、PVMのタスクIDをFOLONプロセスのIDとして、FCPでも利用する。このことで、FCPが管理する情報を簡略化できる。

PVMではプロセスが動的に起動できるが、PVMのタスクIDには各PE内でのプロセスの起動順序情報が含まれている。そこで起動した順序によって、共有メモリ内のバッファの位置を固定することで、テーブルを参照せずにバッファ位置を計算することができる。

さらにMPIでは動的なプロセスの生成は禁じられているので、使用するPE数、共有メモリを使用するプロセス数、割り当てられる共有メモリのサイズが分かれば静的に各プロセスの受信バッファのアドレスを計算することができる。この場合は、各プロセスが属するグループ情報や、同時に起動するMPIプログラム数などはFCPで提供する必要がある。

上記のように上位ライブラリの特性を利用することで、バッファテーブルの廃止、あるいは受信バッファ1

つにつき1DWord☆以内の情報で管理するバッファテーブルによって通信が可能になる。ただしFCP IIのみを使用したプログラムを記述できるように、FCP plus相当のバッファテーブルも使用できるようになっている。

現在、FCP IIおよびそれを用いた, PVM, MPI, FCP plus相当の3種類の通信を実装している。

6.1.2.2 プロトコルの圧縮について

FCP IIでは、現在のFOLONのシステム構成に特化することで、管理すべき情報のサイズを減少させるよう実装している。

	FCP / FCP plus	FCP II
バッファテーブル	4 DWord	1 DWord
受信バッファ管理領域	7 DWord	2 DWord
メッセージヘッダ	3 DWord	2 DWord

表4 FCP/FCP plus と FCP IIのプロトコルオーバヘッド

直接にレイテンシに影響するのは、受信バッファ管理領域とメッセージヘッダの大きさであり、これらの最適化により、FCP IIでは、FCP/FCP plusよりも表5のように、データ転送1回当たりのオーバヘッド(共有メモリに対するロード/ストア)が減る。

転送方向	FCP	FCP plus	FCP II
送信	23 DWord	18 DWord	8 DWord
受信	22 DWord	16 DWord	7 DWord

表5 データ転送時におけるオーバーヘッドの比較

これらにより、FCP IIでのレイテンシは140μs程度となることが予想される。さらに排他処理の最適化でより小さくできると思われる。

6.2 PVMの高速化

今回実装したPVMが、現段階においてまだ十分な性能が得られていない原因としてあげられるのは次の4点である。

- Pvmd-Pvmd間の通信のみFCPを用いている
- Task-Task間の通信が必ずPvmdを経由している
- 今回実装に用いたFCPは、レイテンシ性能のチューニングを行っていない
- PVMの移植時に、UDPをFCPに直接置き換えたため動作が安定しておらず、多少のウェイトを入れる必要がある

以上の問題点に対して、まず第一にPvmd-Pvmd間及びPvmd-Task間の通信手段をより通信性能の優れたFCP plus又はFCP IIに置き換えることにより、PVMの通信性能をあげることが可能である。またPVMにおいては、Task-Task間の通信をPvmdを通さずに直接Task間で通信させることができるので、Task-Task間の通信も同様にFCP plus又はFCP IIを用いることにより、Pvmdを介することによるオーバヘッドを排除することが可能である。

☆ 1DWord=4Byte

7. まとめ

並列計算機システムFOLONへのメッセージパッシングプロトコルの実装と、PVMの移植について述べた。

現在のFCPには種々のオーバーヘッドがあるため、レイテンシがVCLに比べるとかなり大きく、本来のFOLONの性能が発揮できていない。しかし、これまで検討してきた通り、FCPはEthernet上のUDPと比べて、性能向上の余地が大きい。今後はこのFCPのパフォーマンス向上を行う予定である。またPVMについても、デーモン経由の通信のみをFCP上に移植した段階であるので、本来の性能が出ていない。Task-Task間の通信を、直接FCPを用いてできるようにする予定である。

さらに、低レンテンシの共有メモリを持つというFOLONのハードウェア特性を活かし、相互排他の手間と頻度を減らすような、並列ソフトウェア構築法の検討、および並列ライブラリの設計に取り組む予定である。

参考文献

1) BIT3 Computer Corporation: "Bus-to-Bus Adaptor Model 617 Adaptor Hardware Manual", BIT3 Computer Corporation, Minneapolis, Minnesota, 1994.

2) 瀧和男, 小倉毅, 小西健三: "ワークステーション複合体による並列処理システム", 情報処理学会研究報告, 93-PRG-13, pp. 49–56, 1993.

3) Ai Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Mancheck, Vaidy Sunderam: "PVM: Parallel Virtual Machine", The MIT Press, 1994.

4) 上田学, 菊地賢太郎, 土山了士, 小林一成, 大原拓三, 外山孝伸, 上田和紀: "並列計算機システムFOLONの通信ライブラリの設計と評価", 情報処理学会研究報告, 96-ARC-117, pp. 31–36, 1996.

5) 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, pp. 41–48, 1996.

6) Marc Snir, Steve W.Otto, Steven Huss-Lederman, David W. Walker, Jack Dongarra: "MPI: The Complete Reference", The MIT Press, 1996.