ハイパフォーマンス
コンピューティング 77−15
（1999. 8. 3）

クラスタ上のプログラミング開発環境 −SCore クラスタシステムソフトウェア−

堀 敦史† 手塚宏史† 高橋俊行†
住元真司† 曽田哲之††
原田 浩† 石川 裕†

SCore クラスタシステムソフトウェアは，Myrinet を用いたクラスタを対象とした高性能かつスケーラブルな並列プログラミング環境のソフトウェアパッケージである．本稿は，Myrinet 以外のネットワーク，SMP クラスタ，及びクラスタ化されたクラスタという 3 つの新たな形態のクラスタに SCore を対応させる方法について提案するものである．"Composite" と呼ばれる仮想ネットワークデバイスを設け，composite ネットワークデバイスが複数の実ネットワークデバイスとルーティングテーブルを持つことで，これらの形態のクラスタに対応可能であることを示す．ここで提案された方法は，見方を変えれば，ヘテロなネットワーク構成のクラスタへの対応と考えることができる．提案された方法は，現在 SCore 3.0 として開発が進められている．

Programming Environment for Clusters − SCore Cluster System Software −

Atsushi Hori ,† Hiroshi Tezuka ,† Toshiyuki Takahashi ,†
Sinji Sumimoto ,† Noriyuki SodaSRA , Hiroshi Harada †
and Yutaka Ishikawa†

A high performance scalable cluster system software package, SCore, was designed for clusters using Myrinet. To adapt it to a cluster using other networks, an SMP cluster, and a cluster of clusters, the notion of "composite" is proposed in this paper. The "composite" is a virtual network device which consists of a routing table and several physical network devices. From the viewpoint of seamless computing, the "composite" is to handle heterogeneity. New SCore 3.0 is under development for implementing the "composite".

1. はじめに

既にクラスタは並列処理研究者のための研究対象ではなく，幅広い応用を実現する段階に移行しつつある．クラスタ技術をより開花させるためには，クラスタの特長を伸ばし，より幅広いユーザを獲得することである．クラスタの最大の特長としては，*i)* 最新のハードウェア技術を取り込み易いこと，*ii)* ユーザの目的，予算に応じたクラスタを自由に構築できること，*iii)* これまでの市販並列計算機に比べ，価格対性能費に優れていること，*iv)* クラスタの拡張や分割などが比較的容易であること，などが挙げられよう．

SCore クラスタシステムソフトウェアは，Myrinet[1] を用いたクラスタを対象とする統合されたソフトウェアパッケージであり，高効率な並列処理を可能とする[2]．このソフトウェアパッケージは，高性能通信ライブラリ PM[3]，並列オペレーティングシステム SCore-D[4]，マルチスレッド言語 MPC++[5]，標準通信ライブラリ MPICH-PM/CLUMP[6),7] などから構成されている．SCore クラスタシステムソフトウェアは WWW を通じて配布されており☆，既に 200 を越える国内外のサイトからダウンロードされている．

SCore クラスタシステムソフトウェアは改良を重ね，現在バージョン 2.4 となっている．開発を進めるにつれ，いくつかの問題点も明確になってきた．一方，幅広いユーザの意向に応じると同時に最新のハードウェア技術を取り込むためには，根本的な変更が必要であることが判明した．本稿では，SMP を用いたクラスタへの対応，および，クラスタを接続したクラスタという 2

† 新情報処理開発機構つくば研究センタ
Tsukuba Research Center, Real World Computing Partnership
†† (株)SRA

☆ http://www.rwcp.or.jp/lab/pdsoft/dist/

点を取り上げ，これらがMyrinet以外のネットワークへの対応とそれらが「統合されたネットワーク」という考え方で対処可能であることを示す．この考え方はヘテロなネットワークへの対応と考えることもできることを示す．

2. 背景と目的

もともとクラスタ技術とは技術進歩の速いPCのハードウェア技術を素早くキャッチアップすることを目的としている．これは裏返せば，ハードウェアの進歩に合わせてソフトウェアの対応も素早く行なわれなければならないことを意味する．ここで着目するハードウェア技術として，Myrinet以外の高速ネットワークの対応，SMPを用いたクラスタへの対応，および，クラスタを接続したクラスタの3点を取り上げる．

2.1 Myrinet以外のネットワーク

Myrinetは160MB/sという高速なリンク速度を誇り，多段スイッチ構成により比較的大規模なネットワークを構築でき，かつ高いbi-sectionバンド幅の確保が容易なネットワークである．Myrinetの最大の欠点はその価格である．一方，Gigabit-EthernetやVIA[8]などMyrinetに匹敵するリンク速度を発揮するネットワークは，今後急速に普及するものと見込まれ，その結果，低価格化が進むと考えられる．基本的にどのようなネットワークを用いてクラスタを構築するかは，アプリケーションが要求するバンド幅と予算に依存する．

我々はより幅広くSCoreを使ってもらうという立場から，Myrinet以外のネットワークの選択肢を持つべきである，という結論に達した．この結果，我々はGigabit EthernetにPMのプロトコルを移植したGigaE-PM[9),10)]や，UDPプロトコル上にPMプロトコルを載せたPM/UDP[11)]の開発を進めている．

2.2 SMPクラスタ

PCにおいてもSMPの普及が進み，価格対性能費が向上しつつある．SMPをネットワーク接続したSMPクラスタの構築も数多くすすめられている．SMPの性能を引き出すソフトウェア環境としては，pthreadなどのマルチスレッドによるプログラミングや，OpenMP[12)]を用いる方法などが知られている．しかしながら，SMPクラスタの性能を引き出すためのソフトウェア開発環境の整備が成熟していないことが問題となっている．ひとつのPC内のスレッドあるいはプロセス間の通信と，PC間の通信という2つの通信があり，これらをどのようにユーザに見せるべきかが焦点である．

この問題に対し，我々はMPICH-PM/CLUMP[7)]を開発した．これはSMPクラスタにおけいて，ホスト内通信とホスト間通信という2種類の通信をMPIという枠組において，SPMDプログラミングというひとつのプログラミングモデルに統一した環境を提供する．これにより従来のMPIを用いた並列プログラムを変更することなく，SMPクラスタ上で実行することが可能になる．

2.3 クラスタのクラスタ

クラスタの普及が進むにつれ，ひとつの機関が複数のクラスタを保持するようになることも珍しくなくなると考えられる．この場合，複数のクラスタをネットワーク接続し，より大規模な，ひとつの単体のクラスタとして利用したいという状況が発生することが想定される．図1はこのようなクラスタのクラスタ（Cluster Of Clusters，以下COCと記す）の例を示したものである．この例では，クラスタ内はより高速なSAN（System Area Network）を用い，クラスタ間はLANを用いて接続されている．このようにCOCでは複数の異なるネットワークをどう扱うかという問題が生じる可能性がある．

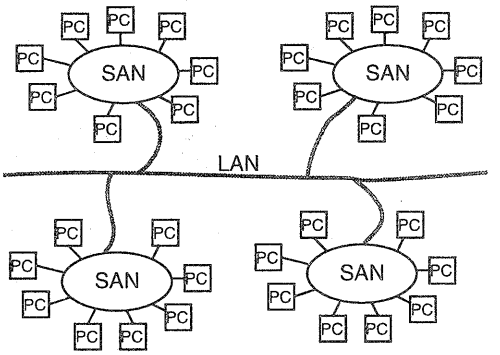

図1 クラスタのクラスタ化

目　　的

本稿は，現在開発が進められているSCore 3.0において，以上に掲げた3つの項目に対応するためにどのような方策を考えているかを述べるものである．そこでは3つの項目それぞれに対し個別に対応するのではなく，「複数ネットワークの統合」という形で統一的に扱えることを示す．

以下，本稿では，まず，このような方策を検討するきっかけとなったMPICH-PM/CLUMPについて概要について述べ，次に複数ネットワークの統合の考え方を示す．

3. MPICH-PM/CLUMP

図2はMPICH-PM/CLUMPにおける通信ネットワークの接続の様子を示している．この図では4CPUを持つSMPのPCが2つ，それぞれMyrinetで接続されている．MPICH-PM/CLUMPでは，ひとつのPCが持つプロセッサの数を上限とするプロセスが複数生成され，プロセス数と同じ数のPMチャネルが

割り当てられる．PM のチャネルとは，ソフトウェア的に多重化されたネットワークのことであり，チャネルをまたがった通信は許されていない．このようにひとつの SMP 内で複数のスレッドでなくプロセスとすることで，複数のスレッド間で共有される変数の排他制御が不要になる．

この図において，PC#0 の Proc#0 から PC#1 の Proc#2 へのメッセージ送信は，PM チャネル #2 に対し，PC#1 を目的地とするメッセージを送信することになる．一方，同じ PC 内の別なプロセスへの通信はプロセス間で共有される System-V の共有メモリセグメントを介したプロセス間通信が行なわれる．

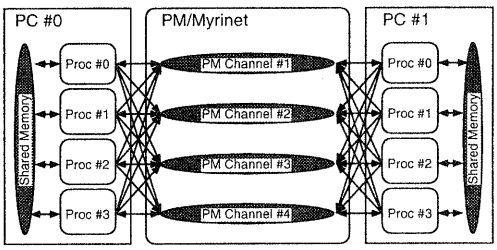

図2 MPICH-PM/CLUMP における通信ネットワークの割当例

ひとつの PM チャネルを複数のプロセスで共有するという実装も可能であるが，こうした場合，*i)* ネットワークから受信したメッセージが自分宛でなかった場合，別途メッセージキューを設け，そこに戻す必要があるためメッセージのコピー回数が増加する，*ii)* メッセージ受信において他のプロセスと競合するため，排他制御が必要になる，という 2 つの欠点がある．

MPICH-PM/CLUMP は，SMP クラスタをシングル CPU のみで構成されるクラスタとしてみせかけるものである．こうすることで，SMP ホスト内通信と，SMP ホスト間通信を区別なく，単一のメッセージ通信としてユーザに見せることができるという利点がある．またこのモデルは，MPI に限らず，全てのメッセージ通信による並列プログラムに適用可能である．

MPICH-PM/CLUMP を用いたプログラムは，現在のところ並列オペレーティングシステム SCore-D の下では動かすことができない．SCore-D では，共有メモリを介した通信を認識していないこと，ひとつのホストに複数のプロセスを生成することに対応していないこと，などといった理由からである．また，SCore-D はネットワークを資源として扱っており，共有メモリによる通信も資源として認識される必要がある．

共有メモリを介したプロセス間通信は，実際にネットワークを通じた通信ではないが，通信の一種であることは間違いない．このように考えると，MPICH-PM/CLUMP は共有メモリを介した通信と，Myrinet を介した通信の 2 種類の通信を，送信の相手先で切替えて行なっていることになる．

図 1 で示したように，COC においては複数の異なるネットワークを持つ可能性がある．これは基本的に MPICH-PM/CLUMP で行なっていることと同じである．もし，複数のネットワークを介した通信が可能ならば，自然と COC への対応も可能になるものと考えられる．さらに SCore-D の実装においても，共有メモリの通信を特別に扱う必要がなくなるという利点も生じる．

4. 複数ネットワークの統合

図 3 は 2 つのクラスタから構成される COC のより具体的な例である．左側のクラスタは 2 CPU 持つ SMP の PC 8 台が Myrinet により接続されており，右側のクラスタはシングル CPU 構成の PC 4 台が Gigabit Ethernet により接続されている．全ての PC は 100Base-T で接続されているものとする．このような COC において，全ての 12 プロセッサを用いる並列プロセスを走らせる場合，ユーザには単一のネットワークが提供されているように見せかけることを考える．

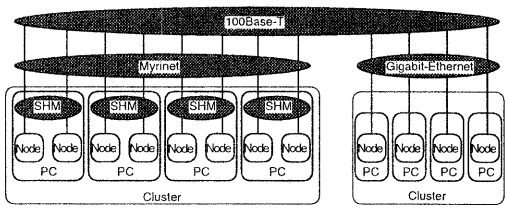

図3 COC ネットワークの例

ここで，MPICH-PM/CLUMP は PM チャネルと共有メモリという複数のネットワークを持ち，相手先のノード番号によりどのネットワーク（チャネル）を用いるかが一意に決まるという点に注目する．

表1 開発中の *PM* デバイス

PM デバイス名	実デバイス／プロトコル
PM(Myrinet)[3]	Myrinet
PM(GNIC-II)[10]	GNIC-II (Gigabit Ethernet)
PM(UDP)[11]	UDP/IP
PM(SHMEM)	System-V shared memory
PM(Composite)	Composite of above devices

以下，*PM* は特定のネットワークの通信ライブラリではなく API を意味するものとする．表 1 に現在開発が進められている *PM* のデバイスを示す．さらに，これらの複数の *PM* をひとつに見せるための仮想デバイス *PM(Composite)* を考える．*PM(Composite)* はルーティングテーブルを持ち，相手先ノード番号から対応する *PM* デバイスとそのデバイスにおけるノード番号を引くことができる．チャネルはそれらのデバイスをオープンしたものであり，一度にオープンできる数はデバイスにより上限がある場合がある．メッセージ

の送信は，結局 *PM(Composite)* のルーティングテーブルを参照し，適切な *PM* デバイスに振り分けられることになる（図4）．一方，メッセージの受信では，*PM(Composite)* が抱えている全ての *PM* デバイスの受信メッセージをポーリングする．SMP における排他制御は *PM(Composite)* が行なう．デッドロックを避けるため *PM* デバイスに対する受信と送信を別なロックで制御する．ひとつしか *PM* デバイスを用いない場合でも *PM(Composite)* を用いることで SMP 対応な *PM* デバイスとすることができる．

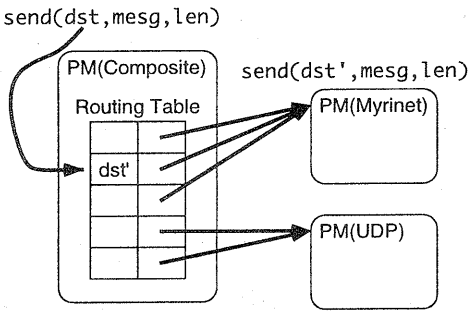

図4 *PM(Composite)* におけるメッセージ送信の例

現時点では，クラスタの全てのホストは並列処理専用とし，ネットワーク間のブリッジあるいはルータの処理は行なわないものとして開発が進められている．しかしながら，例えば 100Base-T のネットワーク内にルータ等が入っていても構わない．そのようなルータの存在はソフトウェアからは見えないからである．このルーティングを認めないという制約と，*PM(Composite)* におけるルーティング方式から，COC においては必ず全てのホストをカバーするネットワークが存在する必要が生じる．

以下，SMP クラスタ上で MPICH-PM/CLUMP に準じた計算モデルを想定した時の，並列 *PM* デバイスの割当ルールと，*PM(Composite)* のルーティングテーブルの具体的な設定方法について述べる．

4.1 *PM* デバイス割当ルール

ここで，ノード番号とは例えば MPI などでいうところの通信のための識別子とし，ホスト名とは *PM(SHMEM)* を除く *PM* デバイスにおける通信のための識別子である．またプロセス番号とは，*PM(SHMEM)* における通信のための識別子を指す．プロセス番号はゼロから順に整数が割り当てられるものとする．

仮定として，あるノードから見た場合，そのノード番号からホスト名とプロセス番号が分かるものとする．また，*PM* デバイスは，そのデバイスを通じて通信可能なホスト名（*PM(SHMEM)* の場合はプロセス番号）を知っているものとする．全ての *PM* デバイスには割当の優先順位が与えられているものとする．

あるノードにどのような *PM* デバイスを割り当てれば良いかは，それぞれのノードにおいて，全ての相手先ノードに対し，以下の規則を適用すれば良い．

- 相手ノードが同じホストの場合，*PM(SHMEM)* をひとつだけ割り当てる．
- 相手ノードが異なるホストの場合，そのノードに通信可能で最も優先順位の高い *PM* デバイスを，相手ノードのプロセス番号に 1 を加えただけ割り当てる（既に割り当てた *PM* デバイス数に不足があれば追加する）．

ネットワークは物理的あるいは経済的な特性から考えて，それがカバーする範囲が狭い程高性能であると考えられる．これに従って，*PM(SHMEM)* > *PM(Myrinet)* > *PM(GNIC-II)* > *PM(UDP)* という優先（性能）順位を与えた場合，先に示した図3における実際の *PM* デバイスの割り当てを示したものが図5である．この図において，例えば *PM(Myrinet)*#0 とあるのは，*PM(Myrinet)* をオープンした 0 番目のチャネルという意味である．結果的に図5では，全体として *PM(SHMEM)* が 1 チャネル，*PM(Myrinet)* が 2 チャネル，*PM(GNIC-II)* が 1 チャネル，そして *PM(UDP)* が 2 チャネル割り当てられている．

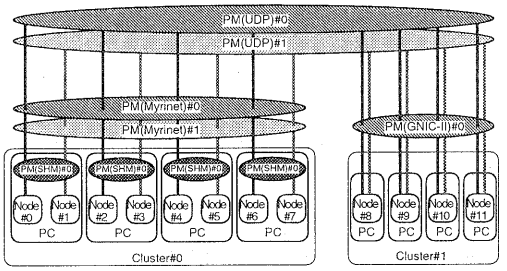

図5 *PM* デバイスの割り当て例

4.2 ルーティングテーブルの生成

PM(Composite) のルーティングテーブルには全ての相手先ノードの情報（*PM* デバイスとそのデバイスにおけるノード番号の対）が必要になる．個々の相手ノードに対するルーティングの設定は以下の規則に従えば良い．

- 同じホストの場合，割り当てられた *PM(SHMEM)* を *PM* デバイスとして設定し，相手先ノードのプロセス番号をノード番号とする．
- 異なるホストの場合，そのホストと通信可能な，相手先プロセス番号目の *PM* デバイスを設定し，そのデバイスにおけるノード番号を設定する．

図3に対応したルーティングテーブルの例を表2に示す．この表では，ネットワークデバイスについてのみ示されている．

表2 ルーティングテーブルの例

SRC \ DST	0	1	2	…	8	9	…
0	-	S_0	M_0	…	U_0	U_0	…
1	S_0	-	M_0	…	U_0	U_0	…
2	M_0	M_1	-	…	U_0	U_0	…
…	…			…			
8	U_0	U_1	U_0	…	-	G_0	…
9	U_0	U_1	U_0	…	G_0	-	…
…	…			…			

S:*PM(SHMEM)*, M:*PM(Myrinet)*
G:*PM(GNIC-II)*, U:*PM(UDP)*

5. 今後の課題

5.1 複数ネットワークの割当

本稿で示した複数ネットワークの割当方式は，基本的に，*i)* ネットワークがカバーする範囲が狭いほど，高性能なネットワークである，*ii)* カバー範囲が狭い高性能なネットワークとカバー範囲が広い低性能なネットワークを組み合わせた場合，できるだけ高性能なネットワークを用いた方がアプリケーションの並列処理効率が高くなる，という考えに基づいている．カバー範囲が狭いが高性能なネットワークをできるだけ用いることで，低性能なネットワークの負荷を減らすことができると推測されるからである．しかしながら，低性能なネットワークがボトルネックとなることも懸念され，実際に高性能なネットワークを併用する意味がどの程度なのかは，評価されるべき項目のひとつである．

5.2 ジョブスケジューリング

SMP クラスタにおいて，投入されたジョブにどのようにプロセッサを割り当てるべきかは未解決な問題である．例えば 4 CPU の SMP クラスタにおいて，あるホストでは 2 CPU しか使われていなかった場合，残りの 2 CPU を他のジョブに割り当てて良いかどうかという問題がある．*PM(SHMEM)* においてネットワークは陽には見えないが，実際にはメモリバスがプロセッサ間通信のネットワークとなっている．また，複数のジョブが Myrinet などの I/O バスにあるネットワークデバイスを共有することも考えられる．結局，ひとつのホストを複数のジョブで共有した場合，資源共有により発生する性能的な相互干渉が発生することになる．もし相互干渉が許容範囲内であるならば，例えば 4 ノードを必要とするジョブを 4 CPU の SMP クラスタに投入する場合，1 CPU × 4 ホスト，2 CPU × 2 ホスト，4 CPU × 1 ホストという 3 通りのスケジューリングが存在することになる．これはスケジューリングの自由度が増えることを意味し，それだけスケジューリングが複雑になる．

COC においては，ネットワークの性能がユーザアプリケーションの性能に影響を与えることが考えられる．また性能の異なるプロセッサで構成される複数のクラスタが COC として構成されることも考えられる．このようにクラスタ上の空間分割スケジューリングによる負荷分散において（例えば[13]），ホストの負荷だけでなく，ホストの違いによる実行イメージの違いや性能の違い，ネットワークの性能の差異も考慮されなければならない．

5.3 SMP クラスタにおけるマルチスレッド

これまで本稿では SMP クラスタにおけるプログラミングは MPICH-PM/CLUMP のモデルを用いて来た．実際にはホスト内でのマルチスレッドプログラミングというモデルも十分に考えられる．本稿では，マルチスレッドの場合，基本的に同一ホスト内のスレッド群は一つの *PM* デバイスを共有するため，*PM(Composite)* が持つ排他制御を導入することでマルチスレッドに対応できると考えている．しかしながら pthread は本来カーネルスレッドであるのに対し，Linux など多くの free な OS ではユーザレベルで実装されている．このために生じる実装上の問題点の有無については現時点で明らかになっていない．

6. シームレスコンピューティングに向けて

本稿で示した *PM(Composite)* による複数ネットワークの統合方式は，シームレスコンピューティングという見方からすれば，異なる複数のネットワークで構成されるクラスタ，および SMP と非 SMP ホストの混成クラスタへの対応と考えることができる．C++ のマルチスレッド拡張である MPC++[5] は既に異なるなプロセッサあるいはオペレーティングシステムで構成されるクラスタ上で既に動作可能となっている[14]．本稿の内容は，シームレスコンピューティング技術[15] のひとつと位置付けられている．

ま と め

PM という統一された通信 API と *PM(Composite)* による複数ネットワークデバイスの統合により，SMP クラスタや COC への対応を統一できることを示した．この方式により，SMP クラスタにおいては，マルチスレッドだけでなく，MPICH-PM/CLUMP が提供する通信モデルもサポート可能であると考えられる．

PM(Composite) による複数かつ異種ネットワークへの対応は，COC を実現するために重要な技術であり，さらには SCore を異機種並列計算環境へ発展させるための 1 段階と考えられている．本稿で述べられた内容は SCore 3.0 として今秋を目処に開発が進められている．評価等については機会を見て発表する予定である．

参 考 文 献

1) Boden, N. J., Cohen, D., Felderman, R. E., Kulawik, A. E., Seitz, C. L., Seizovic, J. N. and Su, W.-K.: Myrinet: A Gigabit-per-Second Lo-

cal Area Network, *IEEE Micro*, Vol. 15, No. 1, pp. 29–36 (1995).
2) Ishikawa, Y., Hori, A., Tezuka, H., Sumimoto, S., Takahashi, T. and Harada, H.: RWCP PC Cluster Programming Environment - Extended Abstract, *Extreme Linux Workshop #2*, USENIX, pp. 16 – 19 (1999).
3) Tezuka, H., Hori, A., Ishikawa, Y. and Sato, M.: PM: An Operating System Coordinated High Performance Communication Library, *High-Performance Computing and Networking* (Bob Hertzberger, P. S.(ed.)), Lecture Notes in Computer Science, Vol. 1225, Springer-Verlag, pp. 708–717 (1997).
4) 堀敦史, 手塚宏史, 石川裕: ギャングスケジューリングの高速化技法の提案, 情報処理学会論文誌, Vol. 40, No. 5, pp. 2072–2083 (1999).
5) Ishikawa, Y., Hori, A., Tezuka, H., Matsuda, M., Konaka, H., Maeda, M., Tomokiyo, T. and Nolte, J.: MPC++, *Parallel Programming Using C++* (Wilson, G. V. and Lu, P.(eds.)), MIT Press, pp. 429–464 (1996).
6) O'Carroll, F., Tezuka, H., Hori, A. and Ishikawa, Y.: The Design and Implementation of Zero Copy MPI Using Commodity Hardware with a High Performance Network, *International Conference on Supercomputing '98*, pp. 243–250 (1998).
7) Takahashi, T., O'Carroll, F., Tezuka, H., Hori, A., Sumimoto, S., Harada, H., Ishikawa, Y. and Beckman, P. H.: Implementation and Evaluation of MPI on an SMP Cluster, *Paralel and Distributed Processing – IPPS/SPDP'99 Workshops*, Lecture Notes in Computer Science, Vol. 1586, Springer-Verlag (1999).
8) Compaq Computer Corp and Intel Corp. and Microsoft Corp: *VI Architecture*. http://www.viarch.org.
9) 住元真司, 堀敦史, 手塚宏史, 原田浩, 高橋俊行, 石川裕: Gigabit Ethernetを用いた高速通信ライブラリの設計と評価, 並列処理シンポジウムJSPP'99, 情報処理学会, pp. 63–70 (1999).
10) 住元真司, 堀敦史, 手塚宏史, 原田浩, 高橋俊行, 石川裕: GigaE PM II: Gigabit Ethernetによる高速通信ライブラリの設計, 情報処理学会研究報告 99-ARC-134 (SWoPP'99) (1999).
11) 曽田哲之, 手塚宏史, 住元真司, 堀敦史, 石川裕: 通信ライブラリ PM の UDP 上への移植と評価, *HOKKE'99*, 情報処理学会, pp. 127–132 (1999).
12) OpenMP Architecture Review Board: *OpenMP*. http://www.openmp.org.
13) 堀敦史, 石川裕, 小中裕喜, 前田宗則, 友清孝志: 時分割空間分割スケジューリング, 情報処理学会論文誌, Vol. 37, No. 7, pp. 1320–1331 (1996).
14) Ishikawa, Y., Hori, A., Tezuka, H., Sumimoto, S., Takahashi, T. and Harada, H.: Parallel C++ Programming System on Cluster of Heterogeneous Computers, *IPPS'99 Heterogeneous Computing Workshop '99*, IEEE, pp. 73 – 82 (1999).
15) 石川, 佐藤, 工藤, 島田: 分散環境におけるシームレス並列コンピューティングシステムの構想, 信学技法 (CPSY), 226, Vol. 97, 電子情報通信学会, pp. 83 – 90 (1997).