ハイパフォーマンスコンピューティング 60－15
（1996. 3. 4）

並列・分散システム性能評価用ツール群 — TEA Library — の設計

関口智嗣[†] 佐藤三久[†] 中田秀基[†]
長嶋雲兵[††] 朴泰祐[†††] 中村宏[†††]

ハイパフォーマンスコンピューティングを目指してこれまで数多くのマルチプロセッサシステムが商用化されてきた．また，ネットワークに接続されたワークステーション上でメッセージパッシングライブラリを用いたワークステーションクラスタシステムでも様々なソフトウエアの実装と実験が行われている．しかし，簡単な並列プログラムを作成するのにも並列プログラムのモデルに関しての知識，通信に関した知識，ライブラリの使用法に関する知識などが要求されている．単純な通信・並列・同期モデルを用いて簡便な並列性の記述，バグの入りにくい記述，簡便な性能測定，容易な移植性と保守性を目指した並列・分散ソフトウエアの開発を目標としたライブラリの提案を行う

The Design of the "TEA Library": A Performance Evaluation Tool-set for Parallel and Distributed Systems

Satoshi Sekiguchi,[†] Mitsuhisa Sato,[†] Hidemoto Nakada,[†]
Umpei Nagashima,[††] Taisuke Boku[†††] and Hiroshi Nakamura[†††]

Today, a bunch of parallel systems has been commercially available in high performance computing world. Various parallel and distributed software for workstaion clusters have been developed as well. Even if one wish to develop rather a smaller software, Programming on various system platforms requires knowledge of parallel programming models, communication models, usage of message passing libraries, and so on. In this article, we propose a library to describe parallel and distributed programs, which allows novice programmers without much effort.

1. はじめに

ハイパフォーマンスコンピューティングを目指してこれまで数多くのマルチプロセッサシステムが商用化されてきた．また，ネットワークに接続されたワークステーション上でメッセージパッシングライブラリを用いたワークステーションクラスタシステムでも様々なソフトウエアの実装と実験が行われている．しかし，簡単な並列プログラムを作成するのにも並列プログラムのモデルに関しての知識，通信に関した知識，ライブラリの使用法に関する知識などが要求されている．例えば，あまり前提となる知識を持たない大学教養学部程度の学生に並列プログラムを作成実習させる場合には，上記の知識獲得にかなりの時間が必要となり，並列アルゴリズムの開発に時間をかけられない．また，中途半端な理解に基づくプログラムでは並列の構造や通信に起因するデバッグが困難となる要因である．

今年度，NAS Parallel Benchmark[1] のサンプルコードを PVM を用いて作成し，それをいくつかの計算機システムに移植することで，それぞれのシステムにおける概略性能を測定することを目標として実装を開始した．しかし，上記のような困難さと，開発者が異なることによるプログラムの不統一さ，保守性の悪さなどに遭遇した．これは，実用に供されているライブラリは必要以上に自由度が高いためで，様々な記述能力を場面に即応して発揮させ，これをプログラムすることが困難であった．

そこで，本稿では並列プログラムの自由度に枠を与えることで容易な構造とし，移植性を高めることで開発基盤（プラットフォーム）に依存しないプログラムの開発と性能測定が可能な並列・分散化のためのライブラリ (TEA: Tools for performance Evaluation and

† 電子技術総合研究所
Electrotechnical Laboratory, AIST, MITI
†† お茶の水女子大学
Ochanomizu University
††† 筑波大学
University of Tsukuba

Analysis) を提案する.

2. TEA 設計目的

例えば現在筆者らが利用可能な並列実験環境の計算機システムを列挙すると, SPARC クラスタ, Indy クラスタ, HP クラスタ, RS6K クラスタ, SPARC マルチスレッド, T3D, C916, J916, T90, SX-3, SX-4, S-3800, VPP500, AP1000, Paragon, CM-5, SR2201, SP-2 等々枚挙にいとまがない. 自由かつ個別のプログラミングをしていると, こうしたシステム間においてソフトウエアを移植することはほぼ絶望的である. 幸なことに, 現在, PVM, MPI, POSIX Thread などの並列記述に関するライブラリの標準化の動きとその実現がなされてきた. しかし, 現時点ではすべてをカバーし, そのままでプログラムを移行できるようなライブラリは存在しない. これは, マルチプロセッサシステムやワークステーションクラスタ等のシステムにおいて開発されたプログラムが

- SPMD, Master-Slave, SIMP, MIMD, マルチスレッドといったプログラムモデルの違い
- 共有メモリか分散メモリというメモリモデルの違い
- PVM[3], MPI[4], P4[2], Express[5], TCGMSG, thread library などの並列・分散化を実現するメッセージパッシングライブラリの違い
- 提供される手段通信方法の違い
- 提供される同期方法の違い
- 基本言語が C または Fortran といった使用言語の違い

といったことに起因する.

一方, 並列システムやワークステーションクラスタ等による分散システムを構築した場合に,

- Ping-Pong や "Hello World" ではなく適度な演算量と通信量を要求するプログラムによる動作テスト
- 基本演算性能, 通信性能の測定
- 他のプラットフォームとの簡便な比較

を実施したいことがしばしば生じる. しかし, 特定のアプリケーションプログラムを用いて測定した場合にはその開発対象プラットフォームに固有の最適化手法, ライブラリの選択, データ配置, 時間測定関数名の違い等を配慮するためしばしば移植性は考慮の外になる. もちろん, そのシステムにおいて最高の性能を求める場合には必須である. しかし, ここでは最高性能追求を求めないと割りきった上で,

- 単純な通信モデル
- 単純な並列モデル
- 単純な同期モデル
- 単純なインターフェース
- 単純な実装方法

を念頭に置いて設計された並列・分散ライブラリを用いることを提案する. このライブラリを用いることにより,

- 簡便な並列性の記述
- バグの入りにくい記述
- 良好な保守性
- 良好な移植性
- 簡便な性能測定

を目指した並列・分散ソフトウエアの開発を目標とする.

表1 並列記述インターフェースの比較 (引数は省略)
Table 1 Various message passing interfaces

Operations	TEA	PVM	MPICH
Create processes	TEA_create	pvm_spawn	MPI_Init
Send messages			MPI_SEND
(Non-Block)	TEA_send	pvm_send	MPI_ISEND
Receive messages	TEA_recv	pvm_recv	MPI_RECV
(Non-Block)		pvm_nrecv	MPI_IRECV
Test buffer status	TEA_test	pvm_nrecv	MPI_Test
Broadcast messages	TEA_bcast	pvm_mcast	MPI_Bcast
Barrier	TEA_barrier	pvm_barrier	
Reduction operation	TEA_reduce		MPI_Reduce
Performance tool	TEA_time		

3. TEA 設計概要

これまでに並列記述の共通性を目指したものとしては代表的なものとして, PVM, MPI などがあげられる.

PVM についてはパブリックドメインソフトであることと様々なプラットフォームに移植されてきたことから事実上の標準になってきている. しかし, PVM はもともとワークステーションクラスタとして設計されてきたことと, 異機種分散処理を強く意識して設計されてきたことなどから機種毎に機能制限があったり, 性能を追求した独自のライブラリが広がったり, 機能変更が頻繁に行われたりした理由で真の標準にはなっていない. また, 集団通信の方法などが十分でなかったりした (表1). さらに, 単純な通信を行うためにも *pack(), unpack()* 関数群や buffer を十分に理解する必要があった.

MPI はもともと PVM, P4, TCGMSG, Express などの共通化を目標としてインターフェースの設計がなされているため, 移植性が優れている. また, MPI で規定されたすべての関数を用いる必要もないためコンパクトなライブラリとすることもできる. しかし, 現状ではまだ十分な実行環境が存在しないこと, 豊富な機能がとっつきにくいこと, マルチスレッドへの対応等の点で十分ではない.

そこで, われわれは以下に述べる制限された環境の下で並列プログラムを開発ならびに評価するために最低限の機能を備えたライブラリを提案する.

- 並列プログラミングモデルとして SPMD のみを採用する. これは適当なマシンで実行形式 (foo) を実

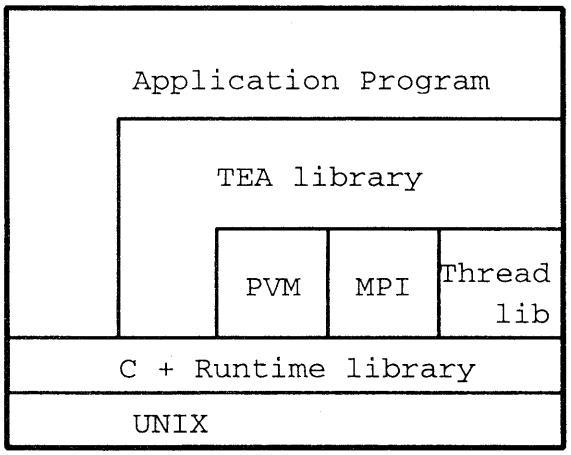

図1 TEA の software architecture
Fig. 1 The software architcture of the TEA

行すると，それが他のマシンにおいて同一名の foo を呼び出すことで並列実行されるプロセスとなる．また，ファイルを1つにできるので保守性が向上する．

- TEA ライブラリは既存の message passing ライブラリの上位に位置し，時間測定用のツール群以外は既存のライブラリを呼び出す（図1）．
- 基本的な関数名等のコンベンションは MPI に準拠する．
- 通信は non-block 送信，block 受信とし状態を得る関数を用意し，dead-lock を避ける（表1）．ブロッキングに関しては送信，受信とも non-block, block の2種類あり，組合わせは4通り考えられる．しかし，non-block 受信は毎回バッファのテストが必要となるので，block 受信とする．また送信は PVM, MPI 等への実装の容易さを考慮し，non-block 送信とした．
- ユーザはアプリケーションプログラムを TEA のインターフェースを用いて記述する．プログラム開発時は可読性を保つため，TEA library とリンクさせる．この TEA library は各種メッセージパッシングライブラリを実際には呼び出すための wrapper として機能し，下記の定義ファイルから作成される．
- デバッグが完了したプログラムは配布のため各並列化ライブラリの定義ファイル (TEA definition) を読み込んで PARMACS 等と同様に m4 macro processor で展開を行うとする．生成するコードは PVM を用いるもの，MPI を用いるもの，POSIX Thread を用いるものの3種類を対象とする．これにより，作成したソフトウエアの配布先でオブジェクトライブラリである libTEA を持つ必要がない．これの様子を図2に示す．
- 時間測定用の関数 TEA_time() を提供する．

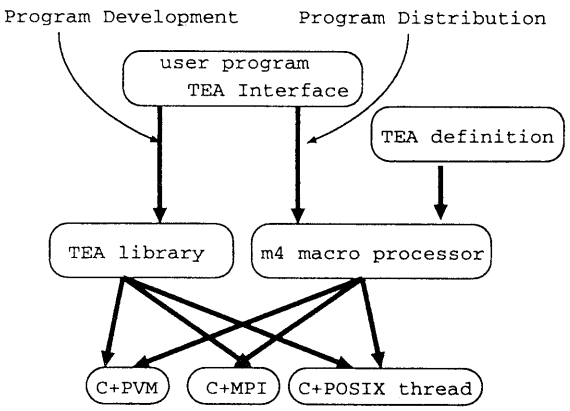

図2 TEA ライブラリの流れ
Fig. 2 The TEA program flow

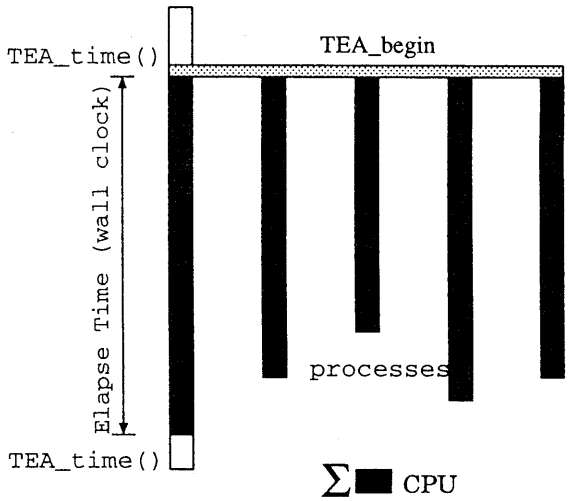

図3 TEA の timer routine
Fig. 3 The timer routine of the TEA

- プロセス間のトポロジは2次元以下とする．これは通信インターフェースを簡略化するのと，プログラムを容易に記述するためである．
- 大域データの分配はファイルを通じて行う．特に，ワークステーションクラスタの場合には中間ファイルを NFS で共有できることを前提としている（図4）．

時間測定に関しては，図3に示すよう，TEA_begin という初期化終了時点からタイマをスタートさせ，この時点からの Elapse 時間と全体のプロセスで消費した CPU 時間の総和を結果として構造体で返す．

$$\frac{\sum \mathrm{CPU}}{ElapseTime \times \mathrm{number\ of\ tasks}}$$

が並列オーバーヘッドに関する効率として定義できる．

TEA は以下の構成要素からなる．

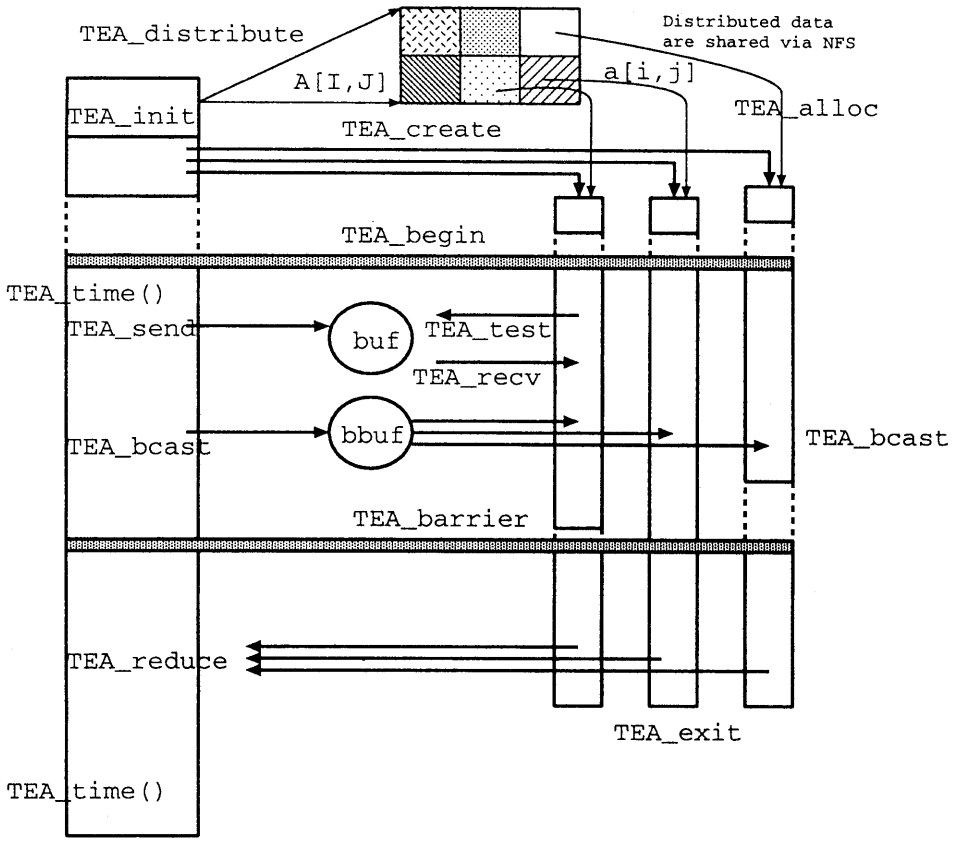

図4 TEA による並列実行
Fig. 4 The parallel execution model with the TEA

- SPMD の立上げ処理，終了処理，メモリマップ等の記述をする **TEA 制御ツール群**
- 大域データを分散メモリに分配する **TEA メモリマップツール群**
- 実際の通信を行う **TEA 個別通信ツール群**
- 集団通信を規定する **TEA 集団通信ツール群**
- プロセスのトポロジを規定する **TEA プロセス構成ツール群**
- 時間測定等の **TEA ユーティリティツール群**
- その他

TEA を用いた並列実行の様子を図4 に図示する．

(1) まず，TEA_init により，初期設定ブロックを記述する．これはシステム全体の初期化を行うところであり，全体で1度しか実行されない．

(2) TEA_create により並列プロセスの生成が行われる．

(3) TEA_distribute と TEA_alloc は対になって，大域変数を分散させてそれぞれのプロセスで保持する．

(4) 全体の並列プロセスが生成された後は，TEA_begin により全体でバリア同期をとる．これにより，すべてのプロセスが実行可能となったことを確認する．

(5) ここで，測定用のタイマーを 0 とする．

(6) データ通信は non-block 型の TEA_send により行う．

(7) データが到着しているかどうかは TEA_test により，buf のフラグを確認するか，block 型の TEA_recv により行う．

(8) 一斉放送は TEA_bcast を用いて行う．これは，全体で受信する必要があるため，受信側も TEA_bcast を用いる．

(9) TEA_barrier はバリア同期を実現する．

(10) TEA_reduce は集団通信として実現され，MPI 準拠の reduction 演算を可能とする．

(11) TEA_exit は生成されたそれぞれのプロセスを消滅させる．

(12) TEA_time によりクロックの値を得る (図3)

4. TEA の応用とその環境

TEA は実際には PVM / MPI / POSIX thread 等

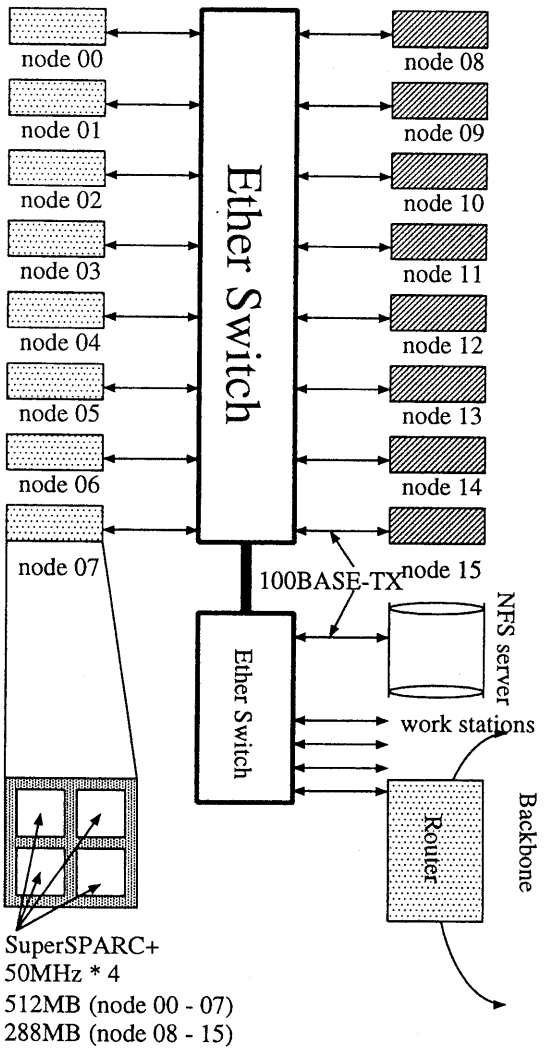

図5 SCoPE 構成の概要
Fig. 5 The overview of the SCoPE

のライブラリに展開されるため，上記のシステムが稼働する環境下では問題なく動作するはずである．しかし，TEA を用いながらワークステーションクラスタにおける可能性を検証してみる．すなわち，これまでのワークステーションクラスタにおける経験[7),8)] から，一般のネットワークに接続されたワークステーションクラスタではその性能評価において外乱要因が多いため結果を十分に吟味できなかった．特に，演算量の少ないプログラムの場合には S/N 比がかなり悪く，測定の度に結果がばらつくこともあった．

そこで，TEA によるソフトウエア開発基盤とし，また評価の基準として図5のようなワークステーションクラスタの専用環境を構築する．

本システムは **SCoPE: Scalable Cluster for Parallel programming Experiments** と呼ぶ．各ノードは SuperSPARC+ の 50 MHz を 4CPU 搭載し，Solaris 2.x の環境下でマルチスレッド実行が可能となる．不必要な daemon は起動しないため，かなりの外乱要因を排除することができる．

各ノードは 100BASE-X の Ether Switch により接続されている．メモリはノード 00–07 は 512MB，08–15 は 288MB 搭載している．これは，当初の目的であった NAS Parallel Benchmark[1)] の実装に必要なメモリ量を検討した結果で決定を行った．表2に文献 6) で用いられたアルゴリズムにおけるメモリの必要量の概算値を示す．縦軸に NPB の各プログラム名とクラスを示す．横軸は複数のタスクに分割した際に，各タスクで必要となるメモリサイズを示す．NPB の場合には実装がプログラマに委ねられているため，アルゴリズムやマッピングが異なるとこれらの数値が異なることに留意して欲しい．表中，ボールドで示してある点は SCoPE でも実装できない問題とサイズである．すなわち，MG の Class A および Class B を 1 台のシステムで実行する場合，FT の Class B を 1 台または 2 台で実行する場合，さらに IS の Class B を 1 台で実行する場合である．プログラムの詳細等に関しては文献 1) を参照されたい．

また，ホストコンピュータノードを設けず，NFS 専用ファイルサーバーを設置することですべてのノードからほぼ均等のアクセス距離を保証する．

専用ワークステーションクラスタの構築と実験が行われているが，SCoPE においてはメモリ搭載量，100BASE-X のネットワークスイッチを用いたこと，各ノードが 4CPU 構成であり，ノードレベルでマルチスレッドプログラムが可能であることなどが特徴的である．特に，プロセスレベルでノードに割り付け，プロセス内はマルチスレッドプログラムを可能とする階層構成はこれを活用して様々な応用が考えられる．

5. おわりに

容易な並列プログラムの作成を目的として TEA Library の設計と提案を行った．その主な特徴としては

- 誰でも，容易でポータブルなプログラムの作成
- 既存のライブラリに対して上位に位置
- 開発用に実行ライブラリを用意
- 配布用に m4 マクロによる展開
- 並列プログラムモデルとして SPMD に限定
- 通信方法の限定
- 時間測定ルーチンを提供

などがあげられる．絶対的な性能では十分ではないが，並列プログラムの動作検証，各種プラットフォームへの移植が容易に行えることが期待されている．現在，詳細なインターフェースの設計途上である．TEA library の実装方法も様々なプラットフォームに m4 マクロで展開するだけではない．MPI が共有メモリの

表2 NAS Parallel Benchmark 実行に必要なメモリ量
Table 2 Memory requirement for NAS Parallel Benchmark

number of tasks	1	2	4	8	16
[EP]					
Sample	-	-	-	-	-
Class A	-	-	-	-	-
Class B	-	-	-	-	-
[MG]					
Sample	10MB	5.2MB	2.7MB	1.4MB	780kB
Class A	**590MB**	290MB	150MB	77MB	39MB
Class B	**590MB**	290MB	150MB	77MB	39MB
[CG (row-mapping)]					
Sample	1.1MB	520kB	270kB	150kB	85kB
ClassA	23MB	12MB	6.0MB	3.1MB	1.7MB
ClassB	170MB	85MB	43MB	23MB	12MB
[FT (fix method, transpose method)]					
Sample	15MB	7.4MB	3.7MB	1.9MB	920kB
ClassA	470MB	240MB	120MB	59MB	30MB
ClassB	**1.9GB**	**940MB**	470MB	240MB	120MB
[IS]					
Sample	1.4MB	680kB	360kB	190kB	110kB
ClassA	170MB	84MB	42MB	21MB	11MB
ClassB	**680MB**	340MB	170MB	84MB	42MB

Multi-Thread programming においても標準となれば TEA library も制限された MPI に展開するだけで十分となる.

TEA で記述されたポータブルなプログラムはそのライブラリキットと共に *http://phase.etl.go.jp/* において公開の予定である.

また, SCoPE はワークステーションクラスタの実証環境として TEA の実装と共に性能評価を行っていく予定である.

謝辞 本研究の一部は科技庁原子力特別研究『複雑現象の解明における超高速計算機利用技術の研究』に基づくものである. NPB に関して詳細な検討をいただいた筑波大学大学院工学研究科板倉憲一氏, お茶の水女子大学理学部情報科学科岩田真由美さんに感謝致します. 研究遂行にあたりご指導いただく電子技術合研究所太田公廣情報アーキテクチャ部長, ならびに日頃より御討論いただく The Hokke-Club メンバー各位に感謝します.

参 考 文 献

1) Bailey, D. H., Barton, J. T., Lansinski, T. and Simon, H.: The NAS Parallel Benchmarks, Technical Report RNR-91-002, NASA Ames Research Center, Moffett Field, CA 94035 (1991).

2) Boyle, J., Butler, R., Disz, T., Glickfeld, B., Lusk, E., Overbeek, R., Patterson, J. and Stevens, R.: *Portable Programs for Parallel Processors*, Holt, Rinehart, and Winston (1987).

3) Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R. and Sunderam, V.: PVM 3.0 User's Guide and Reference Manual, Technical Report ORNL/TM-12187, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831 (1993).

4) Gropp, W., Lusk, E. and Skjellum, A.: *Using MPI*, The MIT Press (1994).

5) Ikudome, K., Kolawa, A. and Flower, J.: 自動並列化システム ASPAR と並列処理実行環境 Express, 電子情報通信学会技術研究報告コンピュータシステム研究会, Vol. 91, No. 130, pp. 213–220 (1991).

6) Itakura, K., Hattori, M., Boku, T., Nakamura, H. and Nakazawa, K.: Preliminary Evaluation of NAS Parallel Benchmarks on CP-PACS, *Proc. of 1995 International Workshop on Computer Performance Measurement and Analysis*, pp. 68–77 (1995).

7) Nagashima, U., Hyugaji, S., Sekiguchi, S., Sato, M. and Hosoya, H.: An experience with super-liner speedup achieved by parallel computing on a workstation cluster: parallel calculation of density of states of large scale cyclic polyacenes, *Parallel Computing*, Vol. 21, pp. 1491–1504 (1995).

8) 坂田聡子, 日向寺祥子, 長島雲兵, 佐藤三久, 関口智嗣, 細矢治夫: ワークステーションクラスタを用いたホモロジー解析, 情報処理学会論文誌, Vol. 36, No. 8, pp. 1987–1994 (1995).