2003－OS－93 (15)
2003／5／9

Suciを用いた高レベル通信ライブラリ

屋比久 友秀[†] 河野 真治[☆,††]

我々は並列分散環境においてユーザーレベルでフロー制御を行う低レベルの通信ライブラリ Suci を実装し評価した．その結果、特定の環境において一般的に利用されている分散通信ライブラリである MPI よりも数倍スループットが向上する事が明らかになった．しかし Suci はメッセージパッシングの基本的な幾つかの機能しか提供していない為、プログラミングが複雑になる傾向があった．この問題を解決する手段として，Suci ライブラリの記述と上位の並列分散記述を結びつける事を考察する．

High level communication library using Suci

Tomohide Yabiku[†] and Shinji Kono[☆,††]

In distributed environment, we have implemented and evaluated a low level communication library “suci” and it control a flow in a user level. Consequently, in some cases, it became clear that the throughput improves several times rather than MPI which is generally used as distributed communication library. However, since Suci has only some fundamental functions of message passing, it had the tendency for programming to become complicated. To solve this problem, we consider description of Suci library, and parallel distribution description of higher rank.

1. はじめに

PC クラスタ上のプログラムは，様々なプログラミング言語によって記述されいる．例えば，Fortran とそれに対するアノテーションを用いた HPF，比較的低レベルな PVM，集合演算に優れた機能を持つ MPI などが良く使われている．我々は，PC クラスタ上のアセンブラに相当するような非常に低レベルな通信ライブラリ Suci を提案してきた．しかし，Suci はポーリングベースのライブラリであるために，メッセージ待ちループを中心とした記述が必要であること，また，信頼性を保証するためのメッセージ再送信も自分で管理する必要があるために，プログラム記述が膨大で複雑になることがわかっている．本論文では，Suci ライブラリの超低レベル (しかしデバイス依存ではない) 記述と，より上位の並列分散記述を結ぶ手法に付いて考察する．

† 琉球大学理工学研究科総合知能工学専攻
Interdisciplinary Intelligent Systems Engineering, Graduate School of Engineering and Science, University of the Ryukyus.
☆ 琉球大学工学部情報工学科
Information Engineering, University of the Ryukyus.
†† 科学技術振興事業団さきがけ研究 21(機能と構成)
PRESTO, Japan Science and Technology Corporation.

2. 背景

PC クラスタなどの並列分散環境において，利用されている通信ライブラリの多くは，トランスポート層に TCP を利用している．これらの TCP ベースの通信ライブラリを用いてプログラミングを行うと特定のアプリケーションでは TCP の性質上、避けられない問題が発生する場合がある．そのアプリケーションの例として、並列タブロー検証系がある．このアプリケーションでは、ランダムな完全結合型の通信が発生し、異なる性質の通信、すなわち応答性重視の通信とスループット重視の通信が発生する．このようなアプリケーションのトランスポート層に TCP を用いて大規模な PC クラスタの環境で完全結合型の通信を行うと，それぞれの通信先毎にソケットをオープンする為，ひとつのプロセスが開けるファイル数の制限やカーネル資源を多大に消費する傾向がある．また，TCP は一様な通信品質になるようにカーネル内部でフロー制御する為，上記のアプリケーションのように通信の優先度が存在する場合は，通信効率が悪くなる．我々はこのような問題を解決する為に，UDP に信頼性を付加した，ユーザレベルでフロー制御を行う通信ライブラリを開発した[1)]．我々はこの通信ライブラリとトランスポート層に TCP を用いた通信ライブラリとの通信速度の比較を行った．その結果，TCP ベースの通信ライブラリよりも数倍高速になる場合があり，特にフロー

制御を並列分散アルゴリズムの一部として扱った場合は通信速度が最大 7 倍まで良くなる事がわかった[2]．我々はこの通信ライブラリを Suci(Simple User level Communication Interface) と名付けた．この Suci はフロー制御をユーザレベルでおこなうため，より低レベルの細かいチューニングが可能である．しかしながら，プログラミングがより複雑になる傾向がある．

我々はこの問題を解決する為に，一般的に利用されている通信ライブラリよりも低レベルな位置に Suci を位置づけ，様々な通信ライブラリで開発された並列プログラムが最終的に Suci レベルに変換される機構を提案する．

3. 通信ライブラリ

この節では，Suci の上位層に位置付けられる一般的に利用されている通信ライブラリと Suci の概要について述べる．

3.1 MPI

MPI はベンダー，ユーザなどの広い範囲での委員会である Message Passing Interface Forum によって，標準として提案された Message Passing library のひとつである．MPI は 1994 年に標準規格の第一版が発表され，1997 年に MPI2.0 が発表された．MPI の実装として代表的なものは，MPICH[5] と LAM/MPI[7][6] がある．MPICH は、クライアント to クライアントモデルにおいて，要求によりコネクションが形成されるのに対して，LAM/MPI では初期状態において全てのネットワークがつながっている．その為，コネクションの立上りにおいて LAM/MPI の方が有利である．MPI の実装は他にも各ベンダーから提供されており，ベンダー独自に MPI を拡張したライブラリも存在する．

3.2 PVM

PVM は TCP/IP ネットワーク環境で接続された計算郡を Message Passing の手法により，仮想並列マシン (単一の並列計算機) として利用可能にした Message Passing library である．異機種混合環境でひとつの分散メモリ型仮想並列計算機として利用できる．プログラム中で PVM の機能を利用する為のユーザインタフェースライブラリと並列プログラムを構成するタスク間の通信，タスク制御を行うデーモンで構成されている．デーモンは，一人のユーザに対してのみサービスを提供するように設計されている．デーモン間の通信は，UDP ソケットを通じて行われている．信頼性のあるパケット配送システムが UDP の上位に実装されている．一方，デーモンとローカルなプロセスの間と同一ホスト内にあるプロセスの直接通信，あるいは異なるホスト間で PVM が直接通信の指示を受けた場合は TCP で通信する．図 1 にその PVM の通信を示す．

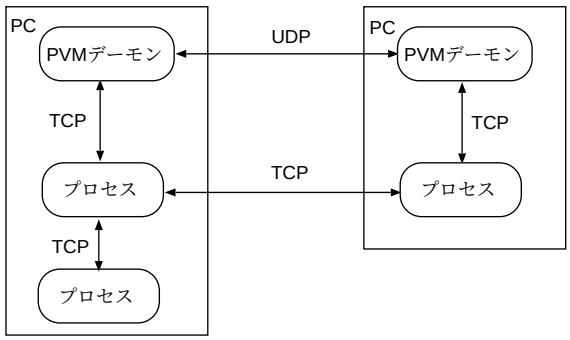

図 1 PVM の通信
Fig. 1 Communication of PVM

3.3 Suci の概要

Suci(Simple User level communication Inteface) は UDP に信頼性を付加した PC クラスタなどの並列分散環境において用いられる事を想定した通信ライブラリである．以下にその特徴を述べる．

(1) ユーザーレベルでフロー制御，輻輳制御が可能．
(2) カーネルの資源消費を最小限におさえる．
(3) Myrinet[3] や Gigabit Ethernet などの特種なデバイスに依存しない．

1 は TCP などのシステムレベルでフロー制御を行う場合は，図 2 左のようにシステム内部にトランスポート層が組み込まれていてユーザレベルであるアプリケーション層から制御できないが，図 2 右のようにフロー制御をアプリケーション層に持ってくる事を意味している．これによって，通信の優先順位があるアルゴリズムにも対応する事ができる．

2 は低レベルの通信レイヤに UDP を使うことで，１つのプロセスが１つのソケットで複数のプロセスと通信することができる．比較的大規模な 60 ノード以上の分散計算環境でも N 対 N の完全結合型ネットワークでカーネル資源をそれ程消費すること無く，構築することができる．

3 は一般的なハードウェアである Ethernet,Fast Ethernet 上での通信を行う事ができ，特種なデバイスを必要としないので，安価で高速な分散計算環境が構築できると考えられる．

Suci はユーザレベルのフロー制御を実現する為に，ユーザメモリ空間に再送用のバッファを持っている．この再送用のバッファは通信相手先ごとに持っている．Suci がユーザ空間に持っているデータ構造と処理の概要を図 3 に示す．Suci は UDP パケットにシーケンス番号を付加し，UDP パケットの欠落や順序の入れ代

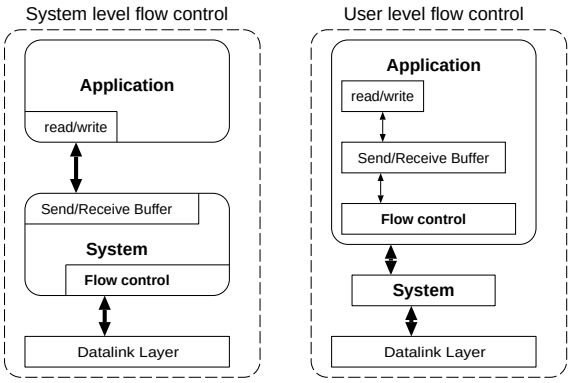

図 2 ユーザレベルフロー制御
Fig. 2 User level flow control

わりが起きても，確認応答と再送要求の処理により，メッセージ配送を保証している．

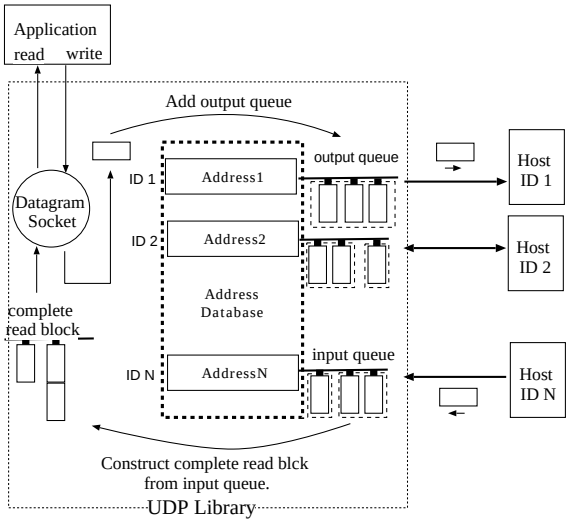

図 3 Suci の構成
Fig. 3 Structure of Suci

4. プログラミングスタイル

この節では，並列プログラミング言語，第 3 節で述べた通信ライブラリのプログラミングスタイル，スケルトン並列プログラミングのスタイルと Suci が想定してるポーリングベースのプログラミングスタイルについて述べ，その比較を行う．

4.1 並列プログラミング言語

通信を含む専用化した並列プログラミング言語を用いてプログラムする方法である．並列プログラミング言語は 80 年代に良く研究されていて，ポートを用いた通信を持つ Occam や，Concurrent Pascal，あるいは，オブジェクト単位で並列実行する並列オブジェクト指向言語，タプルスペースへのタプルの入出力によって通信する Linda などが提案されてきた．しかし，並列プログラム自体が難しいことと，並列プログラミング言語の通信モデルが並列計算機の通信モデルにうまく適合するとは限らないなどによりあまり普及していない．

4.2 逐次型+並列通信ライブラリ

MPI や PVM などで一般的に利用されている並列化手法としては，逐次型のプログラムを書き，それから並列化用の通信 API を埋め込む場合と，HPF(High Performance Fortran) のように逐次型のプログラムを自動的，あるいはアノテーションを埋め込む形で並列化する手法がある．HPF の実装はベンダーなどから多数提供されている．HPF は Fortran のソフトウェア資産やプログラミング経験を生かすことができるという利点があるが，並列分散プログラムの可能性を十分に引き出すためにはアノテーションの付け方やアルゴリズムやデータ構造の変更などに労力をかける必要がある．

図 4 に逐次型のプログラムから並列通信ライブラリを使って並列化する場合の例を示す．これは、配列に格納されている 1〜6 までの数値の合計も求める場合を想定している．このように逐次型から作成されたプログラムはコストの高い部分を他のノードに分散する事で並列化を行っている．

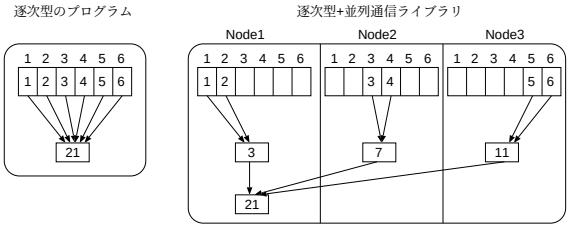

図 4 逐次型からの並列プログラミング
Fig. 4 Paralell programming from sequential program

4.3 並列スケルトンプログラミング

並列マシン上に効率よく実装できる要素を組合せることによりプログラミングを行う スケルトン並列プログラミング[9)10)] が提案されている．スケルトン並列プログラミングの実装としては C 言語のプログラムに並列 API を導入して並列化するものがある[11)]．並列スケルトンのプログラミングは幾つかの並列計算パターンを組み合わせることによってプログラミングを行う．並列計算パターンの種類としては，BMF スケ

ルトン[12)13)14)] などがある．以下にそのデータ並列スケルトンを示す．

- **map スケルトン**
  リストの全ての要素に対して同一関数を適用する．
  map k $[x_1, x_2, \ldots, x_n] = [kx_1, kx_2, \ldots, kx_n]$
- **reduce スケルトン**
  リストの要素に対して結合的な二項演算子を々適用し，ひとつの値を返す．
  reduce $(\oplus)$ $[x_1, x_2, \ldots, x_n] = x_1 \oplus x_2 \oplus \ldots \oplus x_n$
- **scan スケルトン**
  reduce のすべての中間結果を蓄積したリストを返す．ここで，$\iota_\oplus$ は二項演算子の単位元とする．
  scan $(\oplus)$ $[x_1, x_2, \ldots, x_n]$
  $= [\iota_\oplus, x_1, x_1 \oplus x_2, \ldots, x_1 \oplus x_2 \oplus \cdots \oplus x_n]$
- **zip スケルトン**
  2 つのリストの対応する要素を組にしたリストを返す．
  zip $(\oplus)$ $[x_1, x_2, \ldots, x_n]$ $[y_1, y_2, \ldots, y_n]$
  $= [(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)]$

上記 4 つの基本的なスケルトンを組み合わせる事で並列プログラミングを行う．

以下に C 言語によるスケルトン並列プログラミング[11)] の例を示す．利用した API の詳細を以下に示す．

- $[+|-]map < A, B > (f, in, out, size)[+|-]$;
  先頭の [+|−] は，+ は，データ分散有，− はデータ分散無しになる．<> はデータ型を指定する．f は関数を指定し，in は入力データ,out は出力データ,size は入力データのサイズを指定する．末尾の [+|−] は，+ はデータ収集有，− はデータ収集無しを表す．
- $[+|-]reduce < A > (oplus, in, out, size)$;
  先頭の [+|−] は map と同じで，oplus は二項演算式,in は入力データ,out は出力データ，size は入力データのサイズを指定する．reduce の場合は末尾の [+|−] は無い．

```
int square(int n) {
  return n*n;
}

int add(int m, int n) {
  return m+n;
}

int sum_square(int *a, int size) {
  int b[size], result;

  map<int,int>(sqaure, a, b, size)-;
  -reduce<int>(add, b, &result, size);

  return result;
}
```

これは，大きさ size の配列 a の各要素の二乗の和を求めるプログラムで，map スケルトンで配列 a の各要素の二乗を並列に計算し，その結果を b に確保する．ついで，reduce スケルトンで配列 b の全ての要素の和を求めて result に代入し，それを返している．

スケルトン並列プログラムは，このにしてプリミティブなスケルトンを組み合わせる事で並列化の煩わしさを軽減している．基本的なスタンスとしては，逐次型+並列通信ライブラリのプログラミングスタイルをより簡潔に表現しようとしている．そのような意味では，並列通信ライブラリの上位層に位置付けする事ができる．

4.4 ポーリングベースのプログラミングスタイル

Suci の UDP ソケットからの入力 (read) は、select システムコールを用いて非同期に行われる．read できる UDP セグメントがあれば，read され Suci の受信バッファに格納される．read するべき UDP セグメントが届いていなければ、read は行われない．

このような通信イベントの到着を検知する方式をポーリング (polling) という．図 5 がその概要図である．ポーリングベースの非同期入力を使用する利点として，入力 (read) による待ち合わせによっておこるシステムのアイドルを無くすことが出来る．

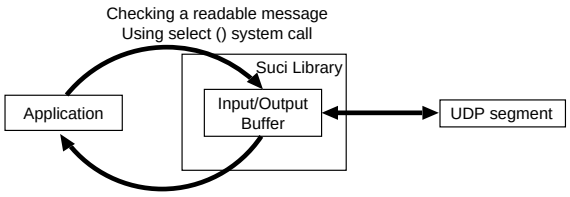

図 5 ポーリングベースのプログラミングスタイル
Fig. 5 Polling based programming style

Suci はこのような基本的にこのようなポーリングベースのプログラミングスタイルを想定しているが，同期機構，フロー制御などはユーザがプログラミングしなくてはならない，MPI や PVM などの API を Suci を使って実装する事も可能である．このように考えると Suci は一般的に利用されている通信ライブラリよりもより低レベルの通信ライブラリだと考える事ができる．

5. Suci を用いた並列スケルトン

第 4 節で見てきたそれぞれのプログラミングスタイルでは，階層がある事がわかる．スケルトン並列プログラミングの下に MPI や PVM などによる逐次型+通信ライブラリの層があり，さらにその下に Suci によるポーリングベースのプログラミングスタイルがあると考えられる図 6.

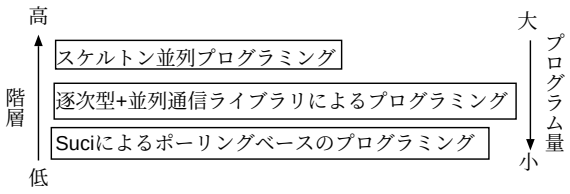

図 6 各プログラミングスタイルの関係
Fig. 6 Relation of each programming style

この関係から，並列スケルトンの map スケルトンと reduce スケルトンを Suci で実装する事が可能であると考えられる．例として 4.3 節で紹介したプログラムの状態遷移図を考察する．

Suci レベルで reduce スケルトンを行う場合は，ひとつのノードに受信データが集中するため，UDP のバッファーオーバーフローが発生する事がわかっている[2]．その為，木構造にデータ収集した方が効率が良い．図 7 に木構造のデータ収集の流れを示す．

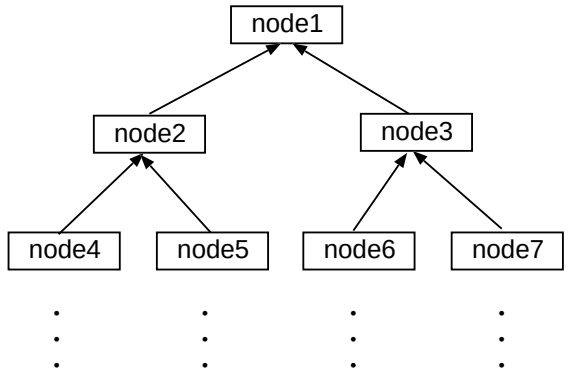

図 7 木構造のデータ収集
Fig. 7 Gather data by tree structure

Suci レベルでは，このようなデータのフローまで考慮したプログラミングが必要になってくる．図 8 に 4.3 節の例題プログラムの Suci レベルの状態遷移図を示す．

6. Suci+Skelton と他の方法の比較

Suci ライブラリは，もちろん，それをそのままプログラミングしても良いが，直接の Suci の使用を避ける方法は，Skelton からコンパイルする方法以外にもいくつかある．

- PVM や MPI のライブラリの実装を Suci ライブラリで記述し，プログラマは PVM,MPI を使用する
- HPF コンパイラの出力として Suci ライブラリを使用する
- 未知の新並列プログラミング言語を Suci ライブラリにコンパイルする

PVM や MPI の実装は TCP を用いたものや Myrinet などを使ったものなどいろいろあるが，Suci を使うことにより UDP の高速性や信頼性の制御などを含むきめ細かい実装を行うことができると考えられる．しかし，PVM や MPI は大量の API を含むために実装の手間はおおきい．また，Suci の利点を生かすためには，実際のアルゴリズムにそってフロー制御などを行う部分を新しく記述する必要がある．

HPF のコンパイル先としても Suci ライブラリは優れている．しかし，HPF の持つ限界を越えることはできない．前にも述べたように，HPF で効率的な並列分散プログラムを行う場合には，アルゴリズムからの変更が必要になる場合が多いからである．また，PVM/MPI の場合と同様に信頼性制御やフロー制御までを含めた最適化を行う場合は，新しいアノテーションの導入などが必要になると考えられる．

新しい並列プログラミング言語を作成すれば，Suci ライブラリの特色を生かした並列プログラミングが可能になると思われる．しかし，例えば，信頼性を保証するための再送制御などを並列プログラミング言語の仕様に含めるとすると並列プログラミング言語自体が低レベルなものになるか，非常に複雑なものになってしまう．

スケルトンの場合は，要求される並列実行が明確なために，そこからフロー制御，信頼性制御，再送制御を含む Suci ライブラリコールの生成が可能になると思われる．また，SIMD 的な PC クラスタ全体での集合演算の繰り返しを越えて，複数の集合演算がパイプライン的に重複したコンパイルをすることも可能である．

しかし，スケルトンの記述があらゆる並列分散プログラムを記述できるほど一般的なものであるとは言えない．Suci ライブラリを用いた低レベルの最適化を含む記述は，スケルトンからのコンパイラに押しつけられることになる．実際には，このような最適化自体が並列分散プログラムの重要な部分であることが多い．実際，スケルトンのコンパイルは様々な並列分散アルゴリズムの選択が含まれ，その時その時の並列計算機の実装技術にも依存する．このようなコンパイラを書くことは不可能ではないが非常に難しいと考えられる．

7. まとめと今後の課題

本論文では，Suci をベースとした場合の上位言語について考察し，スケルトン並列プログラミングを Suci ベースに書き換えた場合の状態遷移図を示した．また，スケルトン並列プログラミング以外にも MPI，PV，HPF などの下位言語に Suci を導入する場合について

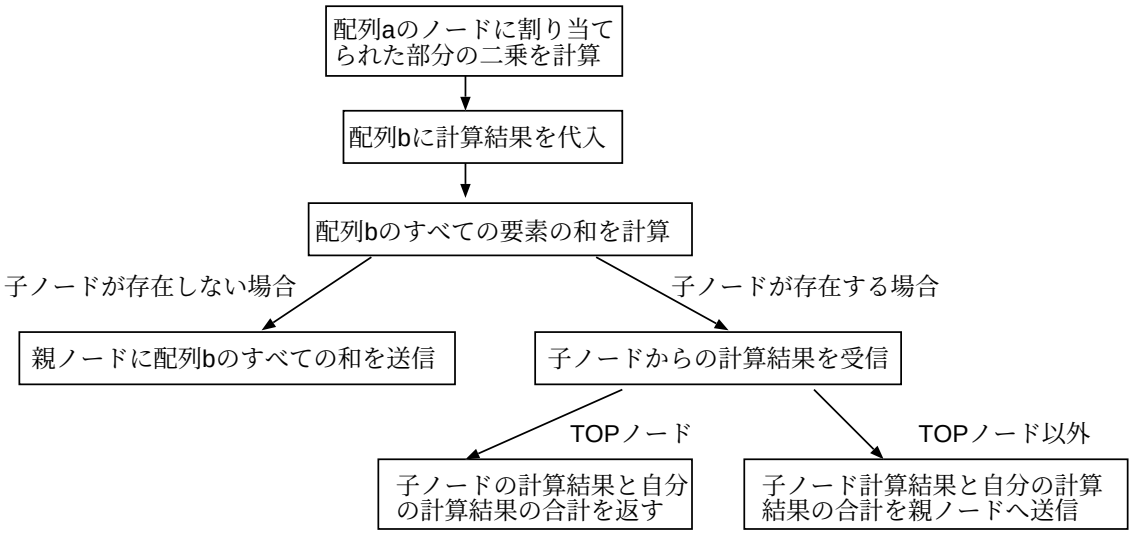

図 8 Suci レベルの状態遷移図
Fig. 8 State transition diagrams of Suci

も考察を行った.

今後は, Suci の上位層として, スケルトンからの簡単なコンパイラを作成し, Suci ライブラリを用いた並列分散プログラムの経験を積むとともに, より柔軟な上位記述と下位記述の連係方法について研究していく予定である.

参 考 文 献

1) 河野真治, 神里健司. UDP を使った分散環境とその応用. 日本ソフトウェア科学会第 16 回大会論文集, 1999

2) 屋比久 友秀, 河野 真治. 並列分散ライブラリ Suci の実装と評価. システムソフトウェアとオペレーティングシステム予稿集,2002

3) Myricom, Inc. http://www.miricom.com/

4) The MPI Forum(http://wwww.mpi-forum.org/). MPI: A Message Passing Interface. *Supercomputing'93*,pages 878-883, November 1993.

5) MPICH. http://www-unix.mcs.anl.gov/mpi/

6) LAM/MPI. http://www.lam-mpi.org/

7) Gregory D. Burns, Raja B. Daoud and James,R. Vaigl. "LAM: An Open Cluster Environment for MPI". *Supercomputing Symposium '94*, June 1994. Toronto, Canada.

8) V.S. Sunderam. PVM: A Framework for Prallel Distributed Computing. *Concurrency: Practice and Experience*, 2(4):315-339, December 1990.

9) M.Cole. *Algorithmic skeltons : a structured approach to the management of parallel computation*. Research Monograps in Parallel and Distributed Compting, Pitman, London, 1989.

10) J.Darlington, A.J. Field, P.G. Harrison, P.H.J.Kelly, D.W.N. Sharp, Q. Wu, and R.L. While. Parallel programing using skelton fuctions. *Parallel Architectures and Languages Europe*. Springer Verlag, June 93.

11) 白沢 楽, 胡 振江, 岩崎 英哉. C 言語上のスケルトン並列プログラミングシステム, 第 5 回プログラミングおよび応用のシステムに関するワークショップ, March, 2002.

12) R. Bird. An introduction to the theory of lists. In M. Broy, editor, *Logic of Programming and Calculi of Discrete Design*, pages 5-24. Springer Verlag, 1987.

13) Z. Hu, H. Iwasaki, and M. Takeichi. Formal derivation of efficient parallel programs by construction of list homomophisms. *ACM Transactions onf Programming Languages and Systems*, 19(3):444-461, 1997.

14) David B. Skillicorn. *Foundations of Paralell Programming*. Cambridge University Press, 1994.