NAS Kernel FT Parallel Benchmark の、PVM による並列解法に関する研究

東海大学　工学部　通信工学科

指導教員　清水　尚彦　講師

研究者　20et3228　松崎　将紀

提出日 平成８年３月８日

目 次

Abstract

近日、NASA は並列コンピュータの性能評価のために、NAS パラレルベンチマークと呼ばれる一連のベンチマークを提供している。本論文ではこのベンチマークの中の、3 次元の FFT からなる問題；カーネル FT に注目し、PVM システム-ネットワークで接続された複数の unix 機上で並列計算をおこなうためのソフトウエアシステム-上に実装し評価した。

実装はイーサネットベースのワークステーションクラスタ上でおこなった。特にボトルネックとなる通信競合の影響を避けるために、(1) 直接メッセージ受信・送信、(2) 通信と演算のオーバーラップ、(3) 通信競合を避けたタスクスケジューリングを適用し評価した。

通信と演算のオーバーラップでは、通信以外の時間のほとんどをオーバーラップできた。また、通信競合を避けたスケジューリングをおこなったことにより、かなりの高速化がおこなえたことを示す。

Chapter 1

はじめに

NASA の提供する NAS パラレルベンチマーク (NPB) は、並列スーパーコンピュータの性能評価のために開発された。

これらの 1 連のベンチマークはカーネルと呼ばれる基本問題 5 題と CFD とよばれる応用問題 3 題からなる。それらは、大規模な航空力学アプリケーションの計算・データの動きをまねたものである。また、データの構造、アルゴリズムの選択、プロセッサとメモリの割当てを実装者が自由に選択できるという特徴を持つ。

本論文では、このベンチマークの中のカーネル FT に着目した。このベンチマークは、3 次元の FFT を用いて偏微分方程式を解いている。

実装は、ネットワークで接続された unix 機のクラスタを、仮想的に 1 台の並列コンピュータとし並列計算を可能とするためのソフトウエアシステム、PVM システムを用い、イーサネットで接続されたワークステーションクラスタ上におこなった。

並列化において通信のオーバーヘッドは無視できないものであるカーネル FT の並列計算では通信は、2-D FFT の後の配列の入れ替えだけであり、基本的に同時に発生する。このような同時に発生する大きな通信は、通信競合を発生させ性能を低下させる。特に、

バスであるイーサーネット上での並列計算においてその影響は深刻な問題である。

本論文では通信のオーバーヘッドの削減方法として、まず直接メッセージ受信・送信、通信と演算のオーバーラップを適用した。直接メッセージ受信は PVM のルーチン psend,precv を用いることで実現した。通信と演算のオーバーラップは 2-D FFT 後の配列の入れ替えと 1-D FFT の間でおこなった。これらの方法を用いることにより、通信以外のほとんどの処理のオーバーラップを可能とした。しかし、実装のモデルは計算にかかる時間にたいして通信が大きいため、通信にかかる時間にオーバーラップさせることができなかった。

また、実装をおこなったネットワークが複数のパーティションに区切られ、パーティション間での通信競合が存在しないという特徴を活かし、通信競合を避けたタスクのスケジューリングをおこない通信オーバーヘッドを削減した。逆方向部分のループを各パーティションに振り分け並列におこなったことで、パーティションごとに全く独立に計算をおこなうことが可能となった。この方法を用いたことで、逆部分の通信を通信競合なしで並列処理でき、かなりの高速化がおこなえた。

Chapter 2

NAS Kernel FT について

2.1 NAS Parallel Benchmark

NPB についての説明を [1] を参考に行う。

2.2 Kernel FT

2.2.1 問題の説明

カーネル FT では、3 次元の DFT を用いて偏微分方程式を解く。以下にその詳細を説明する。まず編微分方程式について考える。

$$\frac{\partial \mu(x,t)}{\partial t} = \alpha \nabla^2 \mu(x,t), x \in R^2, \mu \in C \tag{2.1}$$

$$\mu(x,0) = \mu_0(x) \tag{2.2}$$

ここで x は 3 次元の場の点である。

両辺の 3 次元フーリエ変換の後、この方程式は、

$$\frac{\partial v(z,t)}{\partial t} = -4\alpha\pi^2 |z|^2 v(z,t) \tag{2.3}$$

$$v(z,0) = v_0(z) \tag{2.4}$$

となる。ここで、$v(z)$ は、$\mu_0(x)$ の 3 次元フーリエ変換である。この方程式の解法は

$$v(z,t) = e^{-4\alpha\pi^2|z|^2 t} v(z,0) \tag{2.5}$$

$$= e^{-4\alpha\pi^2|z|^2 t} v_0(z) \tag{2.6}$$

となる。

ここで、離散的な偏微分方程式について考える。以上のステップにしたがい、これは、初期配列$v(x,0)$ の正方向の 3 次元離散フーリエ変換、その結果への指数の乗算、そして逆方向の 3 次元離散フーリエ変換の計算により解く事ができる。$n_1 \times n_2 \times n_3$の配列の正方向の 3 次元離散フーリエ変換と逆方向の 3 次元離散フーリエ変換は、

$$F_{q,r,s}(u) = \sum_{l=0}^{n_3-1} \sum_{k=0}^{n_2-1} \sum_{j=0}^{n_1-1} u_{i,j,k} e^{-2\pi ijq/n_1} e^{-2\pi ikr/n_2} e^{-2\pi ils/n_3} \tag{2.7}$$

$$F^{-1}_{q,r,s}(u) = \frac{1}{n_1 n_2 n_3} \sum_{l=0}^{n_3-1} \sum_{k=0}^{n_2-1} \sum_{j=0}^{n_1-1} u_{j,k,l} e^{2\pi ijq/n_1} e^{2\pi ikr/n_2} e^{2\pi ils/n_3} \tag{2.8}$$

$$\tag{2.9}$$

で定義される。

カーネル FT の計算の流れは以下のようになる。

初期化部分: 指定の乱数発生法を用いて $2 \times n_1 \times n_2 \times n_3$の 64bit の実数を発生させる。そして、連続する２つの実数が、Uの１つの成分に割り当てられるように、$n_1 \times n_2 \times n_3$ の複素数の配列、$U(j,k,l), 0 \le j < n_1, 0 \le k < n_2, 0 \le l < n_3$に割り当てる。

正方向部分: 配列 U の 3 次元 FFT を計算をして、$V(j,k,l)$ を求める。

逆方向部分: for t=1 to N

乗算ステップ: $\alpha = 10^{-6}$として、 $0 \leq j < n_1$,$0 \leq k < n_2$,$0 \leq l < n_3$ での係数配列との乗算 $W(j,k,l) = e^{-4\pi\alpha^2|j'^2+k'^2+l'^2|t}V(j,k,l)$ を行う。ここで、j' は、$0 \leq j < n_1/2$ の時は i 、そして $n_1/2 \leq j < n_1$の時は、$j - n_1$とし、k',l'についても同様に定義される。

逆 3-D FFT ステップ: FFT のアルゴリズムを用いて、配列 W の逆 3 次元 FFT を行い、配列 $X(j,k,l)$ を求める。

チェックサムステップ: チェックサムの計算、$\sum_{i=0}^{1023} X(q,r,s)$ を行う。ここで、$q = i \mod n_1$,$r = 3i \mod n_2$ and $s = 5i \mod n_3$とする。

end for

問題のサイズはクラス A で、$n_1 = 256, n_2 = 256, n_3 = 128$ そして N=6、クラス B では、$n_1 = 512, n_2 = 256, n_3 = 256$、そして N=20 である。3 次元の FFT は航空力学のアプリケーションの重要な部分である。

2.2.2 3-D FFT の説明

カーネル FT の主なタスクである 3 次元の FFT は式で定義される。この最も簡単な解法は 1 次元 FFT の 3 回のくり返しである。つまり、

$$F_{q,r,s}(u) = \sum_{j=0}^{n_1-1} u_{j,k,l} e^{-2\pi ijq/n_1} \tag{2.10}$$

$$F_{q,r,s}(u) = \sum_{k=0}^{n_2-1} u_{j,k,l} e^{-2\pi ikr/n_2} \tag{2.11}$$

$$F_{q,r,s}(u) = \sum_{l=0}^{n_3-1} u_{j,k,l} e^{-2\pi ils/n_3} \tag{2.12}$$

$$\tag{2.13}$$

を順番におこなっていけばよい。

アルゴリズムは以下のようになる。$n_1 \times n_2 \times n_3$ の複素数配列 A があると仮定する。最初に $n_2 n_3$の 1 次元複素数ベクトルでの n_1ポイントの 1 按　FFT 式を実行する。そしてその結果の配列を $n_2 \times n_3 \times n_1$の配列 B に入れ換える。次に B の $n_3 n_1$の 1 次元複素数ベクトルでの n_2ポイントの 1 按　FFT 式を実行する。そしてその結果の配列を $n_3 \times n_1 \times n_2$ の配列 C に入れ換える。最後に C の $n_1 n_2$の 1 次元複素数ベクトルでの n_3ポイントの 1 按

FFT 式を実行する。そしてその結果の配列を $n_1 \times n_2 \times n_3$の配列 D に入れ換える。この配列 D が最終的な 3-D　FFT の結果となる。

Chapter 3

PVM について

PVM についての説明を [2, 3] を参考に行う。

Chapter 4

実装モデルの説明

実装を行うモデルの説明

Chapter 5

問題の並列化

5.1 並列化の方法

カーネル FT の並列化は文献 onPVM,FT,IBM で説明されている。

カーネル FT は、3 次元の FFT の繰り返しである。クラス A では正方向の 3-D FFT を 1 回、逆方向の 3-D FFT を 6 回繰り返し、クラス B では正方向の 3-D FFT を 1 回、逆方向の 3-D FFT を 20 回繰り返す。したがって、カーネル FT の並列化で主となる部分は、3 次元 FFT の効果的な実装である。3 次元の FFT は、1 次元の FFT を 3 つの次元方向それぞれに沿うように 3 回繰り返しておこなう。3 次元配列の j 方向へのの FFT は、

$$F_{q,r,s}(u) = \sum_{j=0}^{n_1-1} u_{j,k,l} e^{-2\pi ijq/n_1} \tag{5.1}$$

で定義される。この式のように、j 方向の FFT には、j 方向に連続する n_1ポイントのデータが必要である。それゆえ、FFT する j 方向の連続するデータが、同一のプロセッサ上に並べられなくてはならない。k れ方向の FFT の場合も同様である。

n_p個のプロセッサでの並列化は、まず、図 1 のように 3 次元のデータを 1 次元方向に沿って、各プロセッサに分割し、それぞれのプロセッサが $n_1 \times n_2 \times n_3/n_p$のデータを持

つように、$n_1 \times n_2 \times n_3$の配列を分割する。それぞれにプロセッサ上に分割された配列でつぎような 2 按　FFT をおこなうことができる。l 方向にデータが分割されたと仮定する。それぞれのプロセッサでは、$n_1 \times n_2 \times n_3/n_p$の複素数配列が組織される。ここでまず、$n_2 \times n_3/p$ の 1 次元複素数ベクトルでの j 方向に沿った n_1ポイントの 1 按　FFT をおこなう。次にその結果を同じプロセッサ上で、複素数配列 $n_2 \times n_3 \times n_3/n_p$に入れ換える。次に $n_3/p \times n_1$の 1 次元複素数ベクトルでの k 方向に沿った n_2ポイントの 1 按　FFT をおこなう。この 2 按　FFT は必要なデータが各プロセッサ上にすべて並んでいるので、他のプロセッサとの通信なしに各プロセッサでローカルに並列におこうことができる。

つぎに l 方向に沿った n_3ポイントの 1 按　FFT をおこなう。しかしこの計算のために必要な n_3の連続したデータは、各プロセッサに分割されていて同一プロセッサ上に存在していない。そのため、2-D FFT の結果の配列を、i 方向に沿って各プロセッサに $n_3 \times n_2 \times n_1$ の配列を分割しそれぞれのプロセッサで $n_3 \times n_2 \times n_1/n_p$の複素数配列を組織する様にプロセッサ間で配列を入れ換えなくてはならない。

この入れ換えでは、各プロセッサの配列をさらに j 方向に分割し各プロセッサと入れ換えなくてはならず。全プロセッサとの通信が必要となる。

最後に配列の入れ換えにより組織された $n_3 \times n_2 \times n_1/p$ の複素数配列を使い、各プロセッサ上で並列に $n_2 \times n_1/p$ の１次元複素数ベクトル上での l 方向に沿った n_3ポイントの 1 按　FFT をおこなう。この FFT は各プロセッサでローカルにおこなえる。

3-D FFT の並列計算における通信は、2-D FFT 後の配列の入れ替えだけである。この通信量は、$n_1 \times n_2 \times n_3 \times (p-1)/p$ の send,recve である。

5.2　解析と問題点

並列計算では通信によるオーバーヘッドが大きく影響してくる。通信オーバーヘッドとしては、ネットワークが通信に必要な時間による遅延や通信の起動、メッセージの送受信なかかる時間があげられる。

ネットーワークがバスである場合、通信路に存在できるメッセージの数はただ 1 つである。このため複数のメッセージを同時に通信しようとする場合通信路競合が発生する。単一バスでは通信路にメッセージが存在している状態時に通信の要求を発生しても、通信はネットーワークが空くまでまたされる。この通信路競合はネットワークの通信能力を悪化させ、通信に必要な時間の影響をさらにおおきくする。

実装をおこなうシステムはすべてイーサーネットで接続されている。イーサネットはバスであるため通信競合が性能に大きく影響する。

このようにカーネル FT は、比較的少ない回数での大きな通信が要求される。

3 按　FFT の並列処理に必要な通信は、2 按　FFT の後の配列の入れ換えだけである。しかしこの配列の入れ換えでは、ほとんどすべてのデータが全データ間で入れ換えられる。

データの分割は全てのプロセッサに平等におこなわれ、全てのプロセッサが同等の計算をおこなうので、本質的に全ての通信は同時に起こる。

したがって通信路の競合が発生しデータの転送にほぼ全部の配列が転送する時間がかかり、実行時間が低下すると予測される。

Chapter 6

オーバーヘッドの削減法

。本論文、通信のオーバヘッドを削減するために以下の 3 つの方法、(1) 直接メッセーッジ送信･受信 (2) 演算と通信のオーバーラップ、(3) 通信の競合を避けたタスクスケージューリングを適用した。

6.1 直接メッセージ受信・送信

PVM におけるメッセージの送信は基本的に 3 つのステージで構成される。まず第一に送信バッファを初期化する。第 2 にバッファにメッセージをパックする。第 3 にメッセージ全体を送信する。メッセージは受信バッファに転送され、メッセージを受信したあとパックされたデータを受信バッファより取り出す。この方法はメッセージをバッファからユーザー領域へコピーする点でオーバヘッドが指摘される。

このドロウバックを克服するものとして、PVM はバージョン 3.2.6 以降に psend precv というルーチンをサポートしている。psend precv はバッファの初期化、パックをせずに、データを直接ユーザ領域からユーザー領域へ転送する機構を持ち、メッセージバッファからユーザー領域にコピーするオーバーヘッドを削減することができる。

直接メッセージ受信が成功するためには条件がある。precv は、メッセージの到着よりも早くプロセッサがメッセージ受信要求を発行した場合のみ、直接ユーザー領域に転送することが可能である。そうでない場合は、メッセージは一旦メッセージバッファに転送され、プロセッサがそのメッセージの受信要求を発行したとき、メッセージがユーザ領域にコピーされる。これでは、コピーのオーバーヘッドは解消されない。

そのため、precv がメッセージ到着より先に発行されるよう注意する必要がある。

しかし、3-FFT の配列の入れ換えでは、送信を各プロセッサが同時に発生し、通信は全プロセッサから全プロセッサの間でおこなわれる。そのため、同じタスクがメッセージ送信後に受信をおこなう場合、メッセージ到着時にタスクは送信が全て終了せずに、、まだ受信要求を発生していない状態にある可能性が高い。PVM のタスクは同時に送信と受信をオーバーラップさせることはできない。

そこで、1 つのプロセッサに受信用と送信用の 2 つのタスクを走らせ、受信するタスクを受信待ちの状態にしておく事で、通信および送信に受信をオーバーラップさせておこなうことを可能とし、直接メッセージ受信を実現した。

プロセッサにタスクを 2 つ走らせる方法としては、PVM によるタスク生成関数;$pvm\_spawn()$ をもちいた。他に unix の機能をもちいてのプロセス複製関数;$fork()$ の使用も検討したが、fork もととなるタスク以外の PVM のタスクとの通信が不可能であるので使用しなかった。

6.2 演算と通信のオーバーラップ

前方法では通信に必要なコピーなどのオーバヘッドは克服されるが、ネットワークが通信をおこなっている時間を待つ、CPU のアイドルタイムを削減することは出来ない。そこで、通信をおこなっている時間に先行できる演算をオーバーラップさせておこなうことにより、通信時間によるアイドルタイムを削減する方法を適用した。

2 次元の FFT の後、1 次元の FFT をおこなうためには、図 2 のように各プロセッサに分割されていた計算に必要な i 軸に沿ったデータを、1 つのプロセッサ上に揃える必要がある。そのためにプロセッサ間での配列の入れ換えがおこなわれる。

この時、i 軸に沿ったデータのある１部分だけでも１プロセッサ上に並べば、その部分だけでの最後の FFT をおこなう事が可能である。つまりある１部分の配列が先に１プロセッサ上に並べば、他の処理に先行してその配列を使っての演算をはじめることができる。

そこでまず、i 軸に沿ったデータが部分ごとに揃うように、2 次元の FFT を何回か分割しておこなう。次に演算が終わった部分から先に送信をおこなうことにより、部分的に先行して配列がそろうようにする。そして揃った部分の配列だけを使った演算を、ほかの部分の通信がおこなわれている時間にオーバラップさせておこなうことにより、通信全てが終了するのを待つアイドルタイムを削減することを可能とした。

これにより、通信を待つアイドルタイム、さらに通信を起動すためのオーバヘッドなどの通信のオーバヘッドを、削減することが可能となる。

また、図 3 のように、１つのプロセッサに入れ替えされるデータをさらにステップ化

して計算し転送することにより、さらに細かく演算をオーバーラップさせることができると考えられる。

しかし、これらの方法ではの入れ替えが一つのプロセッサに集中しておこなわれている。これにより、配列が揃いオーバラップさせた演算をはじめるタイミングに、プロセッサごとに差ができ、１つのプロセッサのタスクだけが先行し他プロセッサ上のタスクの終了を待つといった状態を引き起こすことが指摘される。

この負荷のアンバランスを避けるため、ステップ化した演算・送信を図 4 のような順番でおこい、配列の入れ替えがプロセッサごと均等におこなわれるようにする。これにより演算のオーバラップを各プロセッサで均等におこなうことができる。

図 2 演算のオーバーラップ

図 3 ステップ化した演算のオーバラップ　　図 4 負荷のアンバランスの考慮

6.3　通信競合を避けたスケジューリング

本研究では最終的に、複数のパーティションに区切られたネットワーク上への実装をおこなうつもりである。このネットワークでは、パーティション内では 1 つのメッセージしか存在できないが、それぞれのパーティションで独立に通信を行う事ができる。ただし、異なるパーティションとの間の通信は一つしか存在できない。

逆方向部分のループはループごとに全く独立な演算をおこない、並列に実行してもループ間での通信は全く必要がない。そこで、それぞれのパーティションに、ループを分割するにより通信の衝突を削減する事が可能となる。

Chapter 7

実装結果と解析

7.1 実験

イーサーネットで接続されたNEC EWS4800/320VX、

、その上で以下の解法で並列化した問題を実行しその効果を確認した。

モデル 1 直接メッセージ受信、演算のオーバラップを適用していない普通の方法。

モデル 2 1プロセッサにタスクを2つ走らせることで、直接メッセージ送信・受信の効果、受信のオーバーラップの効果をねらった方法。

モデル 3 タスクを2つ走らせて直接メッセージ受信を行い。プロセッサに全てのデータが揃いしだい演算を開始し、他のプロセッサへの通信と演算をオーバーラップさせておこなう。

モデル 4 図3のように2次元の演算を分割して計算、送信し配列が部分的に揃いしだい演算を他の部分の通信にオーバーラップさせておこなう。

モデル 5 2次元の演算を図4のような順番でおこない演算が終わった部分を送信してい

き、配列が部分的に揃いしだい演算を他の部分の通信にオーバーラップさせておこなう。

モデル 6 ネットワークのそれぞれのパーティションに、逆部分のループを分割して並列に計算しネットワークの競合を避けた。ここでは、ネットワークを 3 つ使いループを 2 回それぞれが実行する方法とネットワーク 2 つにループを 3 回ずつ分割する。

7.2 結果

問題サイズ $128 \times 128 \times 128$ 通信量 210[MB] プロセッサ数 16 台			
	実行時間	直接メッセージ通信	演算のオーバーラップ
モデル 1	668	-	-
モデル 2	592	76/11%	-
モデル 3	497	-	171/25%
モデル 4	476	-	192/29%
モデル 5	470	-	198/30%

表 1　オーバーヘッド削減法の適用とその効果

送信・受信

受信待ち（通信）	計算	WAIT	
78 %	7%	12%	3%

図 5 1プロセッサのタスクの内容と割合

問題サイズ $128 \times 128 \times 128$ 通信量 210[MB]		
プロセッサ数	ネットワーク数	実行時間
16	1	470
32	2	247
48	3	181

表 2　パーティションへの処理の分割

表 1 に $128 \times 128 \times 128$ の問題を 16 台に分割して計算を、直接メッセージ受信と演算のオーバーラップを適用していない場合の結果と直接メッセージ受信と演算のオーバーラップをそれぞれ適用した場合の結果とその効果をしめした。

予測どうり一番効果が現れたのはモデル 5; 負荷のバランスがとれるよう 2 次元の FFT を演算・送信していくもので、最大で 30

図 5 に直接メッセージ受信・演算のオーバラップなしでの実行結果とその割合をしめしたように、通信にかかる時間は約 66%であるので、アイドルタイムと計算のほとんどを通信にかかる時間をオーバーラップできたと考えられる。

直接メッセージ送信・受信と受信がオーバラップされたことにより、通信のオーバーヘッドを 11[%] まで削減できた。

表に逆方向部分のループをネットワークのそれぞれのパーティションに分割し並列実行した結果を示した。実行時間のほとんどが通信のために使われていた時間であるため、

通信の競合の起こらないパーティションに分割したことにより、通信時間がほぼネットワークの数ぶんだけ減り、実行時間の通信はかなり減少された。通信の競合をさけたスケジューリングはかなりの効果をあげた。

メッセージが 1 つしか存在できないイーサネット上での通信は、プロセッサごとの通信が同時に発生するこの問題ではかなりの時間を必要とする。並列化した問題の実行時間のほとんどは通信にかかる時間であり、計算の割合は 18 台数を増やした場合でも、通信量は変化しないので高速化は望めない。

Chapter 8

考察

Chapter 9

まとめ

9.1 結論

9.2 今後の課題

9.3 謝辞

本研究はたくさんの人からの協力により完成することができました。特に、若輩ものの私に、多大なご指導とご助言を頂ました清水尚彦講師に厚くお礼申し上げます。もしこの論文がいささかなりとも評価し得る所があればそれは先生のおかげであります。また研究室の同輩たちにはおしみない協力をいただきましたことを感謝します。

その他、協力をいただきました方々には大変感謝しております。

Bibliography

[1] "THE NAS PARALLEL BENCHNARKS" RNR Technical Report RNR-94-007,March 1994

[2] A.Geist"PVM: Parallel Virtual Machine A User's Guide and Tutorial for Neteorked Parallel Computing"

[3] G.A.Geist"PVM3:Beyond Network Computing" ,LNCS 734 parallel Computation,p194-p203,October 1994

[4] S.Ragsdale "並列処理とオブジェクト指向プログラミング" マグロウヒル,1993

[5] S.White,A.Alund,and V.S.Sunderam, "Performance of the NAS Parallel Benchmarks on PVM Based Networks", Report RNR-94-008 May 1994

[6] R.C.Agarwal,F.G.Gustavson,M.Zubair, "An Efficient Parallel Algorithm for 3-D FFT NAS Parallel Benchmark", Proceeding of SHPCC'94(1994)

[7] R.Agarwal,B.Alpen,L.Garter,F.G.Gustavson,D.J.Klepacki, R.Lawrence M.Zubair,"High-Performance parallel implementation of the NAS kernel benchmarks on the IBM SP2"IBM System Journal 34,No.2,p263-p271,1995

[8] Anshl Gupta,Vipin Kumar,Ahmed Sameh, "Performance and Scalability of Percondition Conjugate Gradient Method on Parallel computers"IEEE trans. Parallel and Distrobuted Systems Vol.6 No.5 MAY 1995

[9] 堀江 健志, 小柳 洋一, 今村 信貴, 林 憲一, 清水 俊幸, 石畑 宏明, "メッセージ通信の分散メモリ型並列計算機性能への影響－通信と演算のオーバーラップと直接メッセージ通信の効果－" 情報処理学会論文誌 April 1994

[10] 紺谷 精一, 佐藤 哲司, "ハイパーキューブにおける通信競合を考慮したスケジューリング法" 電子情報通信学会論文誌'93.7 Vol.J76-D-I No.7

[11] 新濃 清志, 船田 哲男,"数値解析の基礎"

[12] Maria Lucka,"An Effective Algorithm for Computation of Two-Dimensional Fourier Transform for NxM Matrics" ,LNCS 734 parallel Computation,p64-p71,October 1994