ハイパフォーマンス
コンピューティング 52−20
（1994．7．23）

# ハイパクロスバ・ネットワークにおけるNASベンチマークの評価

板倉 憲一、廣野 哲、朴 泰祐、中村 宏、中澤 喜三郎

筑波大学 電子・情報工学系
〒305 つくば市天王台 1-1-1

Tel: 0298-53-6912　Fax: 0298-53-5526
{itakura,hirono,taisuke,nakamura,nakazawa}@arch.is.tsukuba.ac.jp

あらまし

本研究では、プロセッサ間結合網としてハイパクロスバ・ネットワーク (HXB) を用いた超並列計算機において、NAS 並列ベンチマークの中から整数ソート問題 (IS) と FFT 問題 (FT) の評価を行なう。まず、HXB におけるアルゴリズムを示し、次に Processing Unit に汎用 RISC アーキテクチャを想定した時の全処理時間の評価を示す。

HXB は多段クロスバ・ネットワークの一種であり閉塞網であるが、並列処理において現れる各種パターンの転送を高速処理可能な高い柔軟性を持つ。この HXB の柔軟性により、IS と FT におけるデータ転送を高速に処理するアルゴリズムを用いることができることが分かった。

# NAS Parallel Benchmarks Evaluation on Hyper-Crossbar Network

Ken'ichi ITAKURA　Akira HIRONO　Taisuke BOKU
Hiroshi NAKAMURA　Kisaburo NAKAZAWA

Institute of Information Sciences and Electronics
University of Tsukuba
1-1-1 Tennodai, Tsukuba 305

Tel: 0298-53-6912　Fax: 0298-53-5526
{itakura,hirono,taisuke,nakamura,nakazawa}@arch.is.tsukuba.ac.jp

Abstract

In this research, we evaluate two problems in NAS Parallel Benchmarks, IS (Integer Sort) and FT (3-D FFT PDE), on a massively parallel computer. The target machine is constructed with 1024 PU's (Processing Units) connected with Hyper-Crossbar Network. Each PU is equipped with a high-performance RISC processor, cache, local memory and a network interface unit.

First, we show the problem mapping and data transfer algorithms for both problems suitable for Hyper-Crossbar Network, and evaluate the required communication time. Then, we evaluate the total calculation time based on CPU time consumption measured on actual RISC workstations which simulate the behavior of PU's.

While Hyper-Crossbar Network is a kind of blocking multi-stage network, it has very high flexibility and performance for many kinds of data transferring patterns often required on parallel processing. As a result, we can achieve very high speed data transfer in both benchmark problems on the target network architecture.

# 1 はじめに

科学技術計算の分野においては高速な計算機が要求され、これに対して従来のベクトル型スーパコンピュータに代わり超並列計算機がその役割を果たしつつある。特に、PU (Processing Unit) に高性能 RISC プロセッサを多数用い、それらを何らかの結合網で相互結合する方式のいわゆる分散メモリ型アーキテクチャのものが実用化されつつある。

多様な並列アルゴリズムに対応するためには結合網が柔軟であることが要求されるが、なかでもハイパクロスバ・ネットワーク (以下 HXB と省略) は多様な基本転送パターンについて、高い性能を示しており、PU 間の結合網として非常に有力である [3][4]。本報告では、いくつかのベンチマークを HXB 上で処理した場合のデータ転送性能と全体的な処理性能を評価する。

最近の超並列計算機においてはメッシュ結合のものが多く提案されている。この理由は実装の容易性、拡張性にある。隣接転送に関してはメッシュ結合で十分であるが、遠距離あるいは不規則な転送を行なう場合にはメッシュ結合では性能不足である。同様にこれまで並列計算機ネットワークとして注目されてきたものにハイパキューブ・ネットワークがある。ハイパキューブはモデルが数学的に美しく、各種ネットワーク・トポロジをエミュレートでき、また PU 間距離を台数の $log$ のオーダーに抑えることができるため、一時期はそれを用いた並列計算機が多く考案・実装された。しかし、このネットワークは次元数が大きくなると実装が難しく、システム規模が数千台を越えると 10 次元以上の方向にリンクを作らなければならないため、リンクのスループットを上げることが難しく、やはり高性能のシステムには向かないことが明らかになってきた。

HXB[3][5] は、ハイパキューブ・ネットワークの拡張として提案された base $m$-$n$ cube [6] をさらに拡張したものであり、PU 間距離が非常に小さく、また通信チャネル数も多いため、様々なアプリケーションにおける複雑な転送パターンにおいても高い交信性能を保つことができる。また、最近の VLSI 技術の発達により、リンクあたりのバンド幅がメッシュ結合と同程度の中規模クロスバスイッチを用い、次元数を低く抑えることにより、数千台規模のシステムを比較的容易に実装することが可能である。

本報告では、このネットワークを用いた並列計算機の上でいくつかのアプリケーションを処理した時の転送性能を評価する。評価するアプリケーションは NAS ベンチマーク [1] の中から整数ソート問題 (IS) と FFT 問題 (FT) を取り上げる。さらに、PU の代わりにワークステーションを用いて単一 PU の内部処理時間を求め、全処理時間の評価を行なう。

以下の節では評価の対象とする並列計算機、IS と FT のアプリケーションの内容、HXB 上でのアルゴリズム、評価結果を順に示し、考察を述べる。

# 2 ハイパクロスバ・ネットワーク

この節では並列計算機の結合網の一種である、HXB ネットワークについて簡単に述べる。

## 2.1 HXB の構成

3 次元 HXB の例を図 1 に示す。

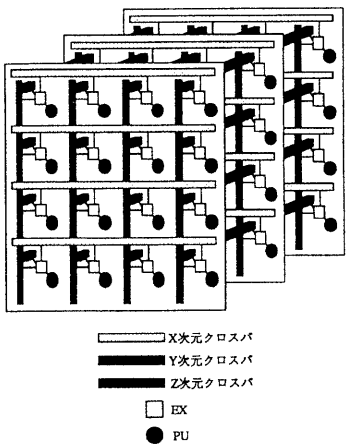


図 1 $4 \times 4 \times 3$ の HXB

3 次元 HXB はプロセッサ (PU) を 3 次元 (各次元を x,y,z とする) 格子状に並べ、各プロセッサは 3 つの次元方向に伸びるクロスバスイッチ (XB) で同一次元方向にある PU と完全結合される。例えば、各次元方向のサイズを $X, Y, Z$ とし、$X \times Y \times Z$ が $8 \times 8 \times 16$ の HXB での各 PU は x 方向で 7 PU,y 方向で 7 PU,z 方向で 15 PU とそれぞれ完全結合される。HXB とほぼ同じトポロジのネットワークに base $m$-$n$ cube ネットワークがある。これは各次元の PU 数が全て $m$ 個で均一のものである。HXB では各次元の PU 数が任意であるため、実装上の構成がより柔軟である。

実際には PU と XB はエクスチェンジャ(EX) と呼ばれる小規模のクロスバスイッチを使って接続されており、これによって 3 つの次元方向の XB を各々結合し、それらの間で wormhole ルーティングを実現する。例えば、PU(0,0,0) が PU(0,3,2) と通信を行なう場合には PU(0,3,0) か、PU(0,0,2) の EX で中継を行なう。

HXB はクロスバスイッチを多段結合した間接網とみなせるが、比較的大きなクロスバスイッチ (例えば 8 入力 8 出力あるいは 16 入力 16 出力程度) を使っているのでネットワークのスイッチング容量は豊富であり、各種の通信パターンに対応できる。更にネットワークのスイッチング容量を有効に活用するために Virtual Channel を用いた適応型ルーティングを用いることも可能である [7]。

### 2.2 想定する並列計算機の構成

基本的に 1024 PU を 8×8×16 の 3 次元 HXB で結合したものを想定するが、参考のために 64 PU (4×4×4), 128 PU (4×4×8), 256 PU (4×8×8), 512 PU (8×8×8) の場合も想定する。ネットワーク上での経路決定は常に X→Y→Z の順にアドレスを決定する (固定ルーティング) とする。

PU は RISC チップを核とし、メモリ、キャッシュ、ネットワークインターフェイスユニット (NIU)、それらを結ぶローカルバスで構成される。動作方式は MIMD 方式で、PU 間でのデータの受渡しは明示的なメッセージパッシングを用いる。また、キャッシュは store-in 方式とし、NIU はバスを用いて DMA 転送を行なう (キャッシュは参照しない) ことにする。このため、転送を行なう際にキャッシュ上のデータをどう扱うかが問題となる。ここでは、明示的なメッセージパッシングを用いるので明示的なキャッシュ命令を用いて制御する。すなわち、送信に際しては、送出するデータがキャッシュにある場合にはメモリに store(フラッシュ) し、受信に際してはキャッシュを無効化してデータはメモリから読むようにする。

## 3 整数ソート問題 (IS)

この節では NAS ベンチマークの中の IS について、その処理の概要と HXB におけるアルゴリズムを述べる。

### 3.1 IS

IS の問題は $2^{19}$bit のデータ値 (key) を持つ $2^{23}$個のデータについて、その順位 (rank) を求めるものである。1 つのデータは 4 つの一様分布乱数の平均であり、その分布は正規分布に近くなる。初期状態では、ランダムに生成されたデータが全 PU に均等にマッピングされ、各 PU のローカルメモリに置かれている。合計 10 回のランを行なうが、各ラン毎に初期データを 2 つずつ修正し、実行結果が異なるようにする。1 回のランが終ると、データ中の特定なもの 5 つについて、rank の検定を行なう。

データの個数に対して key 値の範囲が小さいので、比較型ソートよりも bucket ソートや radix ソートなどの計数型ソートを用いる方が有利である [8]。PU が 1 次元に番号付けされている時の、並列 bucket ソートによる rank の計算手順を以下に示す。

1. local count up

   PU 内の全データに対して、各 key 値のデータ数 (local count) を求める。

2. column rank の計算

   各 PU が自分より若い PU 番号での local count を全て足し合わせ、同じ key 値の中での rank (column rank) を求める。

3. global rank の計算

   最後の PU の column rank は各 key 値の全個数である。この部分和を計算することで各 key 値の base rank を求め、全 PU にブロードキャストする。各 PU は base rank を受けとり、自分の column rank に足し合わせて global rank を計算する。

並列 radix ソートでは、key を数個のビットグループに分割し、各ビットグループ毎に bucket ソートと同様のアルゴリズムを用いて rank を求める。この rank をもとに、実際にデータ転送を行なってビットグループ内でのソートを行なう。データ転送はそのビットグループ内での rank を PU 数で割ることで各データをどの PU のどの位置に転送すべきかを決定する。データ値、元の PU での順番、転送先での位置の情報を、相手 PU ごとにパッキングし転送する。ここでは基本的に全 PU が全 PU に個別のデータを転送することになる (全対全個別転送)。データ転送が済むと、PU 内でデータに対する位置情報を元にして、自分のデータ空間に再構成する。以上の処理を最下位のビットグループから上位に向けて順に繰り返すことによりソートを完了する。

### 3.2 HXB でのアルゴリズムと転送パターン

key 値のビット数と rank 計算テーブルの大きさを考慮して、下位 10bit と上位 9bit の 2 回の bucket ソートによる radix ソートを考える。処理は 10bit の bucket ソートによる下位 rank の計算、データ転送、9bit の bucket ソートによる全体の rank の計算という手順になる。この中で現れる転送パターンが HXB ではどのように扱われるかを述べる。

まず、3次元空間のPUを仮想的な1次元空間へマッピングすることを考える。これは、$X \times Y \times Z$のHXBネットワークにおけるPU$(x, y, z)$のlinear number $l$を

$$l = x \times Y \times Z + y \times Z + z$$

として行なう。

column rankの計算では、全PUでbinary treeを作って1対1の転送を$\log_2 P$回行なう($P = X \cdot Y \cdot Z$)。一辺の個数が2の冪乗のHXBではbinary treeにおけるどの転送も同一次元方向の異なるPU同士の転送となり、無衝突で行なうことができる。また、転送量もlocal rankのテーブルの大きさで決まるので、転送に関係する全PUで等しい。

global rankの計算では、最後のPUによるbroadcastがあり、これはHXBのクロスバスイッチを全開放にすることにより、容易に実現が可能である。この方法のbroadcastはどのPUからでもできるようにするとdeadlockの可能性があるが、単一のPUからのみできるとすれば、deadlockは防げる。

データの並び変えを行なう時には全体全個別転送を行なう。これは先のbroadcastとは異なり、全PUが行なうことと、相手PU毎に転送するデータが違うことが重要となる。3次元HXBでこの転送を行なうには以下のアルゴリズムで、同方向一斉転送を行なう。

```
x, y, z : PU 番号 (0 origin)
X, Y, Z : PU 空間の大きさ (P = X * Y * Z)

PU(x,y,z) において：
for(i=0; i<X; i++)
  for(j=0; j<Y; j++)
    for(k=0; k<Z; k++)
      if !(i==0 && j==0 && k==0)
       PU((x+i)%X, (y+j)%Y, (z+k)%Z) に転送
```

ここで1回の転送を1 phaseとすると、全部で$(P-1)$ phaseの転送がある。各phaseでの転送は全PUで相手PUが異なり、かつ中継のEXも異なるので無衝突となる。しかし、このアルゴリズムで問題となるのは各phaseにおいてPU毎に転送量が違う場合、各phase間での全PUによるバリア同期が必要なことである。もしバリア同期をとらないで上記のアルゴリズムを実行すると無衝突転送は保証されない。また、各phaseで最も転送量の多いPUの転送時間がそのphaseの時間を決定する。フェーズ$\pi$でプロセッサ$p$が必要な転送時間を$T_{p,\pi}$、バリア同期の時間を$S$とすると、全phaseの転送時間$T_{all}$は

$$T_{all} = \sum_{\pi=1}^{P-1} max_p T_{p,\pi} + S \cdot (P-1)$$

となる。全phaseでの総転送量が各PUで同じならば、$T_{p,\pi}$が

$$\forall p \forall \pi T_{p,\pi} = \bar{T}(\text{一定})$$

のときに$T_{all}$が最小となる。ここで転送量の違いを示す尺度としてskew率を

$$\text{skew 率} = \frac{\sum_{\pi=1}^{P-1} max_p T_{p,\pi}}{(P-1) \cdot \bar{T}}$$

と定義すると、skew率は必ず1以上になる。そして

$$T_{all} = (P-1) \cdot \text{skew 率} \cdot \bar{T} + S \cdot (P-1)$$

となり、skew率は1が最適値で、これが大きくなるほど転送に時間がかかることになる。

# 4 FFT問題 (FT)

この節ではNASベンチマークの中のFTについて、その処理の概要とHXBにおけるアルゴリズムを述べる。

## 4.1 FT

FTは、FFT(高速フーリエ変換)を用いて偏微分方程式を数値的に解く問題である。問題空間は3次元であり、この空間上の点を3次元FFT処理し、ある係数を掛けた後に、3次元逆FFT処理を行なうことによって、任意の時刻の状態を調べることができる。解くべき偏微分方程式は

$$\frac{\partial u(x,t)}{\partial t} = \alpha \nabla^2 u(x,t)$$

であり(ただし$x$は3次元空間の座標)、このフーリエ変換は

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

となる。さらに、この一階偏微分方程式は

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

として解くことができ、これを逆フーリエ変換すれば、元の空間での解が得られる。問題空間は$256 \times 256 \times 128$点の複素数で与えられる。一様乱数により$t = 0$の状態を定義し3次元FFT処理を行う。その後、6通りの$t$に対する$e^{-4\alpha\pi^2|z|^2 t}$の値を各点について計算し、逆3次元FFT処理を行なう。各$t$に関して$u(x,t)$が求まると、そのうちの1024点のデータを集計し、checksumを計算する。

## 4.2 HXB での処理方式

このベンチマークにおいては、3 次元 FFT および 3 次元逆 FFT の処理をどのように並列化して行なうかが問題となる。多次元 FFT は、1 次元 FFT を次元数分、毎回結果に対して行なっていく。よって、3 次元 FFT は逐次に 1 次元 FFT を 3 回行なう。FFT を並列計算機で行なう典型的な例は途中のバタフライ転送をネットワークを使って行なうものである。ここでは、1024 PU ($8 \times 8 \times 16$) の 3 次元の HXB において、上の 3 次元 FFT を行なうアルゴリズムとして、PU の担当するデータ空間が固定のもの (固定方式) と、PU の担当するデータ空間を転置操作によって変えるもの (転置方式) の 2 種類を提案する。より少ない PU 台数の場合も同様のアルゴリズムを用いる。

1. 固定方式

   3 次元の PU 空間に 3 次元の問題空間を直接マッピングし、1PU には $32 \times 32 \times 8$ 点を持たせる。そして、例えば X 方向の FFT は、Y×Z 本のデータ列を同時に処理する。

   各次元方向の FFT は、PU 間の 2 点でのバタフライ演算と PU 内の 2 点のバタフライ演算に分けられる。PU 間の 2 点でのバタフライ演算はバタフライ転送を PU 間で行ない、相互に持っている全てのデータを交換転送する。PU と問題空間の X 軸, Y 軸,Z 軸を入れ変えてマッピングを行なった場合でも、必要なバタフライ転送の回数は変わらず、全 PU による $log_2 P$の回の転送が必要であり、1024 PU では 10 回である。

2. 転置方式

   X、Y、Z の次元方向の順に FFT を行なうと仮定する。X 次元方向の FFT は 256 点のデータ列が $256 \times 128$ 本あり、これらは独立に処理を行なうことができる。この並列性ををうまく使うために 3 次元空間の PU を仮想 2 次元平面に並べ、最初に FFT を行なう次元方向 (X) のデータを全て 1PU にマッピングし、1PU には $256 \times 8 \times 4$ のデータを持たせる。各 PU は $8 \times 4 = 32$ 本の 256 点ずつの X 次元データに対して、1 次元 FFT を PU 内処理だけで行ない、その後で次の Y 次元方向が 1PU にマッピングされるように転置転送を行なう。3 次元の問題空間を a、PU 内の 1 次元データ配列を d とすると、転置転送は

   $$\mathrm{a}(x,y,z) : \mathrm{d}[x]\mathrm{PU}(y,z) \rightarrow \mathrm{a}(y,z,x) : \mathrm{d}[y]\mathrm{PU}(z,x)$$

   となる。以上の 1 次元 FFT と転置転送を 3 回繰り返して、全次元に関する FFT を行なう。

上の 2 つの方式を転送コストと内部処理のコストの点で比較する。まず、固定方式のバタフライ転送は FFT 処理を行なっている次元方向にある PU 間でデータ転送が行なわれ、これは同じ XB に接続されている PU 間の通信となるので無衝突転送である。1 回の転送量は 1 PU に割り当てられる点の全データあり、1024 PU で行なった場合、1 回の転送量は 128 Kbyte となり、これは一般に転送の立ち上げ overhead を十分に無視できる長さである。つまり、ネットワークの転送コストは全転送量によって決まり、1024 PU の時の全転送量は 128 Mbyte となる。

これに対して、転置方式の転置転送は 2 次元 HXB に 3 次元データをマッピングしているときには無衝突で行なえることがわかっている [9]。そのアルゴリズムは

```
x, y : PU 番号 (0 origin)
A, A : PU 空間の大きさ (P = A * A)

PU(x,y) において：
for(t=0; t<A; t++){
  PU(y, (x+y+t)%A) に送信
  PU((x-y-t)%A , x) から受信
}
```

である。これを 3 次元 HXB で行なうには、3 次元 HXB を仮想 2 次元 HXB として用いる。具体的には $8 \times 8 \times 16$ PU を 仮想 $32 \times 32$ PU に展開することを考えると、PU $(x,y,z)$ と 仮想 2 次元 HXB で PU $(m,n)$ の対応は、IS で考えた linear number $l$をもちいて

$$x \times 8 \times 16 + y \times 16 + z = l = m \times 32 + n$$

とする。この仮想 2 次元 HXB を用いて、上のアルゴリズムを使うと、無衝突で転送できることを確認した。

1024 PU では、1 回の転置は 32 回の転送に分けられ、1 PU の持っているデータをすべて他の PU に転送するので、1 回の転送量は 4 Kbyte になる。これも立ち上げ overhead を無視できる程度の長さなので、ネットワークの転送時間は全転送量で決まる。3 次元 FFT 処理を行なうためには 3 回転置を行なうので 1024 PU の全転送量は 384Kbyte となる。よって、ネットワークの転送コストは転置方式の方が少ないことが分かる。

さらにプロセッサの内部処理時間の点から考えると、固定方式の PU 間の点に対しては処理の粒度が細かくなり、FFT 演算を行なう時の半分のデータが NIU を経由してメモリにおかれるためキャッシュを有効に使うことができない。これに対して、転置方式では 1 つの次元方向の FFT を行なっている間は PU 数によらず最大 256 点のデータを繰り返し使うのでキャッシュを用

いて処理することができる。これらから内部処理のコストも転置方式の方が少ないことが分かる。

以上のことからHXBでは転置方式の方が適しているのでこの方式を用いて評価を行なう。

# 5 評価結果と考察

HXB並列計算機において、NASベンチマークのISとFTの処理時間を示し、評価結果を検討する。

## 5.1 評価の方法

評価の方法は、ネットワークの転送時間とPU内部の処理時間を分けて考え、転送と処理のオーバラップは考えない。転送のためのデータのパッキング/アンパッキングの時間はPU内部処理時間として評価する。

ネットワークの転送時間は前節までで述べたネットワークの転送パターンを考慮し、転送量と1回の転送立ち上げオーバヘッドから机上計算で求める。想定したネットワークの諸元を表1に示す。

表1 ネットワークの諸元

| | |
|---|---|
| 転送立ち上げ over-head | 5$\mu$sec |
| 転送 through-put | 200Mbyte/sec |

転送スループットが最大性能の半分になる時の転送データ長のことを$N\frac{1}{2}$という。表1の仮定から$N\frac{1}{2} = 1K$byteとなる。

PUの処理時間は、並列処理を行なうプログラムから単一PUの動作を抜き出し、それを実際にワークステーション(HP 9000/735 : PA-RISC 1.1, 99 MHz, 256 KB データキャッシュ)上で実行し求めた。プログラムはC言語で記述し、最適化はCのソースレベルで行なった。具体的にはキャッシュのラインコンフリクトを避けるようなデータのlocation、PU台数、データ数等をソース中に定数記述をし、最適化コンパイル(HP-UX 9.0,cc -O)したものを用いる。また、2.2で述べた転送時のキャッシュとNIAの干渉を処理する状況をシミュレートするために、並列プログラムで転送を行なう部分にキャッシュをクリアする命令列を挿入した。用いた並列プログラムの正当性は、PVM (Parallel Virtual Machine)を用いて実際に並列処理を行って確認した。

## 5.2 ISの結果

ISの転送は10bitもしくは9bitのテーブルの転送と全対全個別転送の2種類に分けられる。

まず、実際のベンチマークにおける乱数列でrankを計算し、全対全個別転送を行なった時のskew率の評価をした。その結果を表2に示す。

表2 ISの全体全個別転送のskew率

| PU数 | 64 | 128 | 256 | 512 | 1024 |
|---|---|---|---|---|---|
| skew率(1) | 1.06 | 1.14 | 1.31 | 1.68 | 2.57 |
| skew率(2) | 1.28 | 1.42 | 1.88 | 3.29 | 7.18 |

skew率(1)は下位10bitでbucketソートを行った結果に対して全対全個別転送を行なう時のものであり、skew率(2)は上位9bitでbucketソートをした結果に対して全対全個別転送を行なう時のものである。ISで求められている処理では、skew率(2)の転送は実際には行なわないが、もし全PUを通してデータがソートされて再配置されるような問題では必要となるので参考として調べた。PU台数が多くなると担当するデータ数が少なくなるので、転送元から見ると転送先毎のデータ数のバラツキが大きくなり、各phaseでの転送量のバラツキも多い。結果的にskew率が大きくなる。

skew率(2)がskew率(1)に比べて大きい理由は以下の通りである。skew率(1)は正規分布の下位10bitでrankを求めているが、この10bitの値の分布はほぼ一様になるので、比較的1に近い値となっている。これに対して、skew率(2)は上位9bitでrankを求めているので、その分布は正規分布に近くなり、中央付近に対応するPUへの転送は特定のPUからのみ行なわれるので、skew率が高くなる。

skew率を含めた全体全個別転送の評価を表3に示す。

表3 ISの全体全個別転送の評価

| PU数 | 転送回数 | 転送量 (byte/回) | 全転送時間 (sec) |
|---|---|---|---|
| 64 | 63 | 26K | 0.0134 |
| 128 | 127 | 7K | 0.0175 |
| 256 | 255 | 2K | 0.0256 |
| 512 | 511 | 640 | 0.0421 |
| 1024 | 1023 | 240 | 0.0764 |

PU台数が増えるにしたがって全体として必要な転送量は減り、転送の粒度が細かくなる。512 PUでは$N\frac{1}{2}$を下回り、転送の回数が$O(P)$で増える。この結果、転送の立ち上げoverheadが転送時間の半分以上を占め、転送量の割には多くの時間がかかってしまっている。

これに対して、10bit, 9bitのテーブルの転送は4Kbyte (4byte×1024個)もしくは2Kbyte (4byte×512個)のデータ長で、転送回数は$O(logP)$である。PU数の増加に対して、転送回数が$O(logP)$で増えるが転送量が$N\frac{1}{2}$程度なので、転送時間の増加は$O(logP)$と考えられる。

ISの全処理時間とその内訳を表4に示す。表4におけるexecはPU内部処理時間、transはデータ転送時間、totalは全処理時間を示す。内部処理時間に関し

てはPUが多くなるにつれてワーキングセットサイズが小さくなり、local count upの時間、パッキング/アンパッキングの時間が$O(1/P)$で減るので線形に減少する。全処理を見ると512 PUまでは台数効果によって性能が上がっているが、1024 PUではデータ転送がoverheadになり性能向上の割合が落ちている。64 PUの処理時間を基準とした台数効果のグラフを図2に示す。この図からも分かるように、512 PUまでは良い台数効果が出ているが、1024 PUでは、ネットワークによる転送がoverheadとなり、性能向上に頭打ちになりかけている。概して、ISのベンチマークは、千台以上のプロセッサを持つ並列計算機にとって、高い台数効果をだすことは難しい。

表4　ISの処理時間(単位はsec)

| PU数 | exec | trans (%) | total |
|---|---|---|---|
| 64 | 2.84 | 0.0177 ( 0.62) | 2.86 |
| 128 | 1.36 | 0.0222 ( 1.61) | 1.38 |
| 256 | 0.64 | 0.0308 ( 4.60) | 0.67 |
| 512 | 0.29 | 0.0478 (14.06) | 0.34 |
| 1024 | 0.14 | 0.0826 (37.55) | 0.22 |

なお、参考値として他の並列機による結果を表aに示す[2]。他の並列機と比較すると、高速に処理されていることが分かる。この理由はradixソートの欠点であるデータ転送をHXBの柔軟性によって補い、全体として高速に処理が行なえるからである。

ISのような非数値処理の場合には一般に内部処理に対してデータ転送量が多いので、転送ネックとなり易い。しかし、並列ソートにおいては実際のデータを移動させることをせずに、keyと元のデータへのindexのみの転送で行なうことが可能であり、今回のISの評価もそのような処理をしている。つまり、単純なデータ転送量の増加によって転送ネックになることはない。

## 5.3　FTの結果

転置方式で行なったFTの結果について述べる。このアルゴリズムは、縦と横サイズが等しい仮想2次元HXBを想定するため1024 PU, 256 PU, 64 PUの評価結果のみを示す。

表5　FTにおける1回の転置転送の評価

| PU数 | 転送回数 | 転送量 (byte/回) | 全転送時間 (sec) |
|---|---|---|---|
| 64 | 8 | 260K | .01053 |
| 256 | 16 | 32K | .00270 |
| 1024 | 32 | 4K | .00081 |

転送のパターンは4.2で述べたように仮想2次元HXBにおける転置転送であり、3次元HXBにおいて無衝突の転送である。転置転送の評価を表5に示す。各回の転送量は$N\frac{1}{2}$を越えており、転送時間は全転送量によって決まる。総転送量は1 PUに割り当てられるデータ量に等しく、PU台数が増加するにしたがって、これは$O(1/P)$で少なくなる。FTの問題サイズでは1024 PUにおいても1回の転送量が$N\frac{1}{2}$を越えるが、さらに大きなサイズのHXBにおいては$N\frac{1}{2}$を下回ることが容易に予想される。この場合、$O(P^{1/2})$の転送回数に対応する立ち上げoverheadが全転送時間のほとんどを占め、全処理時間に占める転送時間の割合が増加する。

次にFTの全処理時間とその内訳を表6に示す。

表6　FTの処理時間(単位はsec)

| PU数 | exec | trans (%) | total |
|---|---|---|---|
| 64 | 9.44 | 0.220 (2.28) | 9.66 |
| 256 | 2.30 | 0.057 (2.42) | 2.36 |
| 1024 | 0.41 | 0.017 (4.07) | 0.43 |

内部処理に関しては、実行時間の大部分が1次元FFT処理とパッキング/アンパッキングの時間である。1次元FFT処理を行なう部分はPU台数に関わらず、キャッシュ内に収まるワーキングセットサイズであり、実行時間は1次元FFTを行なう回数に比例する。1024 PUが256 PUに比べて5倍以上速くなっている理由は、1 PUの担当する点のデータが256 PUでは512 KBであるが、1024 PUでは128 KBなので全てのデータがキャッシュに入る。このため、データ受信後のアンパッキングの処理によって全てキャッシュに入り、1次元FFTの処理とパッキングが高速になるためである。今回は固定方式では評価を行なっていないが、4.2でも述べたようにワーキングセットと転送量から転置方式のほうが高速である。

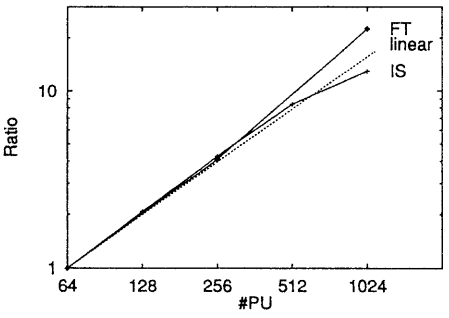


図2　64 PUを1としたspeedup率

64 PUを処理時間を基準とした台数効果のグラフを図2に示す。全処理時間としては、データ転送時間と

内部処理時間が共に $O(1/P)$ になるので高い台数効果が得られた。これは仮想2次元HXBの転置転送を3次元HXBがエミュレートでき、多次元FFT処理の並列性をうまく活かしたアルゴリズムを使うことができたからである。

なお、参考値として他の並列機による結果を表a [2]に示す。いま、T3Dの処理時間が全てFFTの時間であったとしても、この時間に対するHXBの転送時間は数%であり、ネットワークは高い性能を示していることが分かる。これに対して、1次元FFTを1PUで行なう処理については、更に高速化の余地があり、これが、他の並列機よりも性能のおとる主原因と考えられる。

## 6 まとめと今後の課題

PU間結合網としてHXBを用いた並列計算機においてNASベンチマークのISとFTを評価した。転送性能の評価は転送パターン、転送回数、転送量から転送時間として求めた。さらにワークステーションを用いてPUの動作をシュミレートし、実測することで全処理時間を評価した。

HXBは隣接転送のみではなく、広範囲にわたる転送に関して良い特性を持っている。このことを利用し、ISとFTの両方に対して無衝突の転送パターンで処理を行なうアルゴリズムが考えられた。

ISにおいてはradixソートを用いたがその中での全対全個別転送で問題となる転送の偏りをskew率(最適な転送に対する比率)でモデル化した。1024 PUで下位bitのskew率は2.57、上位bitにおけるskew率は7.18となる。高いskew率は転送性能に大きく影響するが、ISの評価で用いたアルゴリズムでは上位bitの転送は必要がなく、高い転送性能を保つことができた。

FTにおいては3次元HXBで2次元HXBをエミュレートでき、転置転送が高速に行なえるので、高い台数効果を生むことができた。これは、全処理時間の内、転送にかかる時間は1024 PUの時で4.07 %に抑えること、及び転置転送することでPU数によらず、1次元FFTはキャッシュを用いて行なうことができるために、内部処理時間が高速に行なえることの2点による。

今後の課題としては、残りのNASベンチマークを評価することと、転送と内部処理のオーバラップを考えた処理方式での評価をとることが挙げられる。

## 謝辞

本研究に関し貴重な御意見を頂いた、筑波大学西川博昭助教授ならびに中澤研究室並列処理研究グループ諸氏に深く感謝します。なお、本研究の一部は文部省科学研究費(奨励(A)06780228)及び創成的基礎研究費(06NP0601)の補助による。

## 参考文献

[1] David Bailey et al. "The NAS Parallel Benchmarks ", Report RNR-91-002 Revision 2, 22 August 1991

[2] David H. Bailey et al. "NAS Parallel Benchmark Results Update", RNR Technical Report RNR-94-006, March 21, 1994

[3] 斉藤 哲也 他, "超並列計算機のネットワークの実現可能性と性能評価", 情処研報 92-ARC-95, 1992年

[4] 石橋 規子 他, "大規模並列処理ネットワークにおけるランダム転送性能の評価", 第46回情処全大 予稿集, 1993年

[5] 田中 輝雄 他, "高並列計算機による空気力学シミュレーションの構想", 第8回航空機計算空気力学シンポジウム論文集, 1990年

[6] 鈴岡 節 他, "並列AIマシンProdigyの相互結合網の評価", 信学論 D Vol.J71-D, No.8, 1988年

[7] 朴 泰祐 他, "ハイパクロスバ・ネットワークにおける転送性能向上のための手法とその評価", 並列処理シンポジウム JSPP '94 pp. 129-136

[8] 板倉 憲一 他, "ハイパクロスバ・ネットワークにおける並列ソート処理", 第47回情処全大 予稿集, 1993年

[9] 斉藤 哲也 他, "ハイパ・クロスバ・ネットワークを用いた並列計算機におけるアレイの高速転置アルゴリズム", 第45回情処全大 予稿集, 1992年

表a Cral Y-MP(1 PU)との並列機(64 PU)によるISとFTの実行結果。

| Computer System | IS (sec) | FT (sec) |
|---|---|---|
| Cray Y-MP | 11.46 | 28.77 |
| Cray T3D | 3.42 | 3.28 |
| IBM SP-1 | 3.06 | 6.46 |
| Intel iPSC/860 | 17.3 | 20.9 |
| Intel Paragon (OSF1.2) | 4.34 | 9.1 |
| Intel Paragon (SunMos) | 3.77 | 7.2 |
| Kendall Square KSR1 | 6.6 | 9.2 |
| nCUBE-2S | 23.2 | 62.8 |
| Thinking Machines CM-5 | 24.2 | 7.9 |
| Thinking Machines CM-5E | 3.1 | 3.9 |

([2]より抜粋)