ハイパフォーマンス
コンピューティング 55－7
（１９９５．３．９）

超並列計算機 CP-PACS における NAS-PB の仮想評価

板倉 憲一、服部 正樹、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: 0298-53-5206
{itakura,hattori,taisuke,nakamura,nakazawa}@arch.is.tsukuba.ac.jp

あらまし

超並列計算機 CP-PACS はノードプロセッサに擬似ベクトルプロセッサ PVP-SW を用い、プロセッサ間結合網には HyperCrossbar-Network (HXB) を用いた数千台規模の並列計算機である。現在 CP-PACS は開発中であり、個々の要素である PVP-SW や HXB の性能評価は行なわれているが、現実的なアプリケーションによる性能評価を行なう必要がある。

本研究では NAS Parallel Benchmarks に定義されている 5 つのカーネル問題を用い、現実的なアプリケーション に対する CP-PACS の性能を仮想評価し、CP-PACS における並列化の手法について考察する。CPU 時間の評価としては PVP-SW 化したプログラムを専用シミュレータ上で実行し所要クロック数を計測する。ネットワーク転送時間の評価としては HXB 用に最適化したデータ転送パターンを机上解析し求める。

評価の結果、64 PU 構成時に関しては、MG 及び CG において高い性能が得られ、また FT に関しては PU 数を増やした時にスーパリニア的な特性が得られることがわかった。逆に CG に関しては PU 数を増やしていくと、ネットワーク転送時間が増加し、性能が低下することがわかった。この問題に対する対応策についても検討する。

NAS Parallel Benchmarks Evaluation on CP-PACS

Ken'ichi ITAKURA　Masaki HATTORI　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: 0298-53-5206
{itakura,hattori,taisuke,nakamura,nakazawa}@arch.is.tsukuba.ac.jp

Abstract

CP-PACS, a massively parallel processor, is equipped with thousands of node processors using PVP-SW (pseudo vector processor) and Hyper-Crossbar Network (HXB) as interconnection network among nodes. Currently, CP-PACS is under construction, and we need to evaluate the total performance of CP-PACS applying many scientific problems.

In this research, we evaluate the total performance of CP-PACS based on five kernels in NAS Parallel Benchmarks, and consider about their parallelization techniques for CP-PACS. CPU time is measured by a specified simulator for PVP-SW. Network data transferring time is estimated from several performance specification according to the data transferring pattern on HXB.

As a result of evaluation, CP-PACS with 64 PU's achieves an excellent performance for kernels MG and CG, and some superlinear effect is caused in kernel FT increasing the number of PU's. On the other hand in kernel CG, the performance is degraded by the increasing of data transferring time when the number of PU's increased. We also consider the solution to this problem.

1 はじめに

本報告では、千台規模のプロセッサによる超並列計算機 CP-PACS (Computational Physics by Parallel Array Computer System 、以下 CP-PACS) [9] の仮想性能評価を行なう。CP-PACS は現在筑波大学を中心に開発が進められており、計算物理学の分野における大規模科学技術計算を主たる目的とする。CP-PACS はノードプロセッサ (以下 PU) に PVP-SW (Pseudo Vector Processor based on Slide-Windowed Registers) [5][8] を用い、PU 間を Hyper Crossbar Network (以下 HXB) [10] で結合した分散メモリ型超並列計算機である。PVP-SW ではキャッシュに頼らないメモリレイテンシ隠蔽の手法を、既存ののアーキテクチャと上位互換性を保ちながら、レジスタへの preload 機能によって提供し、擬似的なベクトル処理を可能にする。また、PU 間結合網の HXB は柔軟性が非常に高く、転送性能を低下させることなく多様な PU 空間を提供することができる [3] 。

CP-PACS の性能評価には NAS Parallel Benchmarks [2] のカーネル問題を用いる。このベンチマークは複数のアプリケーションで構成されており、完全並列な問題、multi-grid 法を用いたポアソン方程式の求解、CG 法を用いた固有値問題、多次元 FFT 問題、並列ソート問題の 5 つからなる。これらの問題は基本的な解法が与えらるが、並列化の手法は各並列計算機に適したものを考えて良く、CP-PACS においても個々の問題に対して、HXB の柔軟性と PVP-SW での擬似ベクトル処理を考慮し、データのマッピング、転送パターンなどの点から最適な並列化について考察する。

各アプリケーションの評価は、CPU 時間とデータ転送時間に分けて行い、CPU 時間は クロックレベルの厳密なプロセッサ・シミュレーションによって求め、データ転送時間は用いる転送パターンとネットワークの諸元から机上計算によって求める。

2 超並列計算機 CP-PACS

CP-PACS [9] の目標最高性能は 300 GFLOPS であり、最低でも 1024 台の PU を結合することを前提としている。CP-PACS の構成図を図 1 に示す。

CP-PACS の PU は プロセッサの他に主記憶、キャッシュ、ネットワークインターフェイスアダプタ (以下 NIA)、記憶制御装置 (SCU : Storage Control Unit) から構成される。そして、プログラム及びデータは各 PU の主記憶に置き、独立にプログラムの実行を行う MIMD 動作方式を用いる。PU 間でのデータの受渡しは、ユーザプログラム中に明示的なメッセージパッシングを起動する命令を置いて実現する。CP-PACS のアーキテクチャの大きな特徴は プロセッサに PVP-SW を用い、PU 間を HXB によって結合することである。

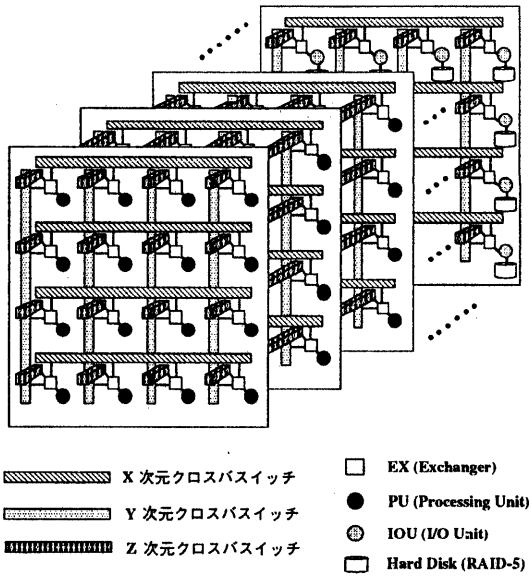

図 1: CP-PACS の構成図

PVP-SW [5][8] はキャッシュを用いずにメモリレイテンシを隠蔽する方法を提供し、擬似的なベクトル処理を可能にする。メモリレイテンシの隠蔽は命令の完了を待たずに後続の命令を処理することが可能なレジスタへの prefetch (preload 命令) によって行なう。さらに、データがメモリからレジスタに届くまでの時間に十分な演算を行なうためにレジスタ数を拡張し、この拡張したレジスタはレジスタ・ウィンドウを用いてアクセスする。ベクトル処理のためのループの演算が終了すると、ウィンドウを前方にスライドさせる。データは preload 命令によって先のウィンドウのレジスタへ load しておき、ウィンドウがその位置まで移動した段階で処理される。

CP-PACS における データ転送は PU のメモリ間で NIA による DMA 転送を用いて行なうが [9] 、この時にキャッシュとメモリの整合性が問題となる。CP-PACS では自動的に整合性をとりながら転送するモードも用意するが、この場合、転送 throughput が低下する。これに対して、ユーザが明示的なキャッシュコントロール命令を用い、整合性はユーザ責任の下で高速転送するモードも用意される。PVP-SW ではキャッシュを介さずに直接メモリを参照するので、preload 命令によってのみ参照されるメモリ領域に関しては、キャッシュ制御命令を使うことなくユーザ責任の下に高速転送が可能となりる。このように PVP-SW は並列計算機と相性が非

常に良いといえる。さらに、転送の立ち上げ overhead を軽減するために、NIA の起動を OS を介さずにユーザが直接行なえるようにもなっている。

次に、CP-PACS で用いる 3 次元 HXB [10] について述べる。3 次元 HXB は図 1 に示したように 3 次元正方格子状に PU を並べ、その間をクロスバスイッチ (XB) で結合した間接網である。各次元方向 の PU は直接 XB によって完全結合され、それ以外の PU 間のデータ転送はエクスチェンジャー (EX) と呼ばれる 小型の クロスバスイッチによって 複数次元方向の XB を経由して行なう。CP-PACS ではメッセージ転送に wormhole ルーティング を用い、経路の選択はデッドロックを防ぐために固定ルーティングを用いる。以下に HXB の特徴を示す。

- 基本転送に関するコストパフォーマンスが良い。
- ランダム転送時の性能が高い。
- 別構成の HXB がエミュレート可能である。
- 任意サイズの正方格子結合 (mesh/torus) のネットワークがエミュレート可能である。
- ハイパキューブのエミュレートが可能である。
- broadcast 転送をハードウェアで行なえる。

一般に隣接転送だけで解くことのできる科学技術計算は多いが、それ以外に global な転送を行なう問題に対しても HXB は対応可能である。さらに、3 次元 HXB の柔軟性により自由な論理的 PU 空間を ユーザに提供することができる。例えば、3 次元 HXB をそのまま 3 次元 PU 空間として用いることもできるが、1 次元 PU 空間や 2 次元 PU 空間として用いること可能である。3 次元 HXB(サイズ $X \times Y \times Z$) の座標 (i, j, k) と 1 次元 (サイズ $L = X \times Y \times Z$) の座標 (l) の相互変換は以下のように行なえる。

$$l = i \times Y \times Z + j \times Z + k$$

また、2 次元 (サイズ $G \times H = X \times Y \times Z$) の座標 (g, h) は 1 次元 (サイズ $L = X \times Y \times Z$) の座標 (l) を用いて、以下のように相互変換できる。

$$l = g \times H + h$$

この時に、論理的な 1 次元 PU 空間や 2 次元 PU 空間においても様々な転送が無衝突で行なえる。

本論文では基本的に 1024 PU を $8 \times 8 \times 16$ 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 ($4 \times 4 \times 4$), 128 PU ($4 \times 4 \times 8$), 256 PU ($4 \times 8 \times 8$), 512 PU ($8 \times 8 \times 8$) の場合についても性能評価を行なう。

3 NAS Parallel Benchmarks

NAS Parallel Benchmarks [2] (以下 NASPB) は、科学技術計算を中心とした並列計算機のためのベンチマーク問題であり、5 つのカーネルプログラム (表 1) から構成される。各問題について、2 つの問題サイズ (Class A, Class B) があるが、ここでは問題規模の小さい Class A を評価する。

表 1: NASPB のカーネルプログラム

名前	内容
EP	乗算合同法による乱数生成
MG	multi-grid 法によるポアソン方程式の求解
CG	CG 法による大規模疎行列の最小固有値問題
FT	多次元 FFT による 3 次元偏微分方程式の求解
IS	大規模な並列整数ソート問題

NASPB の特徴は問題がソースプログラムではなく、アルゴリズムによって与えられる点である。現在、並列計算機は様々なアーキテクチャを持っており、プログラミングパラダイムも各並列計算機毎に異なっている。このような状況の下では、ソースプログラムによって提供されるベンチマークは対象とする計算機において並列処理できない可能性があり (例えば、並列化コンパイラが使用できない等)、不適当である。NASPB では問題の規模、初期データ、解法のアルゴリズムを与え、各計算機毎で並列化を自由に行なうことにより、その計算機の実質的な性能を評価することができる。

NASPB の基本ルールを以下に示す。

- Fortran-90 (Fortran-77 を含む) か C 言語を基本とした高級言語を用いる。
- Fortran と C 言語 の混用はできない。
- 浮動小数点演算は全て倍精度 (64bit) で行なう。
- ベンダーから提供される、すべてのユーザが使用可能なサブルーチンや関数は用いて良い。

アセンブラなどの低水準言語の使用は禁止されているが、CP-PACS の重要な特徴の一つである PVP-SW 機能に対応したコンパイラは現在開発中であり、NASPB のプログラムをそのままコンパイルできる状況ではない。そこで、本研究では 基本的に C 言語 を用いたプログラミングを行なうが、PVP-SW 機能を用いるルーチンは アセンブラによって記述する。

4 評価の方法

評価の方法は、まず CP-PACS のアーキテクチャを考慮した並列プログラムを作成し、 PVM (Parallel Vir-

tual Machine) [4] を用いて実際に並列動作させ、NASPB の要求する結果に正しく合うことを確認する。

次に、各ベンチマークの時間計測を、ネットワークの転送時間と CPU 処理時間に分けて行なう。この際、転送と処理のオーバラップは考えないことにする。ネットワークの転送時間はネットワークの throughput と 1 回のメッセージ転送のための 立ち上げ overhead 、転送量と転送回数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 4 に示す。転送量と 1 回の転送立ち上げ overhead の関係より、転送 throughtput 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。今回の評価の仮定では $N_{\frac{1}{2}} = 1K$byte となる。

表 2: ネットワークの諸元

転送立ち上げ over-head	5μsec
転送 through-put	200Mbyte/sec

CPU 処理時間は、PU に使用するプロセッサのシミュレータにより評価する。まず、並列処理を行なうプログラムから、単一 PU の動作を抜き出した C 言語のプログラムを作成し、CP-PACS のプロセッサのベースとなる PA-RISC 1.1 を用いたワークステーション (HP 9000/735) の最適化コンパイラによりコンパイルする。PVP-SW の機能を用いるルーチンに関しては、アセンブラでプログラムを書き直し、コンパイルした他のルーチンとリンクし実行イメージのオブジェクトを得る。この実行イメージのオブジェクトをシミュレータ上で実行・解析し、実行クロック数を求める。想定したプロセッサの諸元及び仮定を以下に示す。

- 2 命令同時発行の superscalar
  (整数演算 2, 浮動小数点演算 1, load/store 命令 1 の中から 2 つを同時発行可能)
- 1 命令で乗算と加算を行なう浮動小数点命令があり、理論 peak 性能は 2flopc (FLOating Operation Per Clock: 1 クロック当りの浮動小数点演算数)
- 整数演算レイテンシは 1MC (Machine Cycle)
- 浮動小数点演算レイテンシは 3MC
- 主記憶は Interleaved Multi-Bank Memory (16 bank 構成)
- 主記憶アクセスレイテンシは 40MC (この内 bank busy time は 16MC)
- プロセッサの clock は 100MHz
- データ・キャッシュ は 256KB (direct map、ブロックサイズは 32B、ストアイン方式)
- 命令キャッシュ は all hit

なお、今回想定した諸元は、CP-PACS の初期設計段階の値であり、実際のシステムの性能は若干修正される可能性がある。

5 FT と IS

FT と IS に関してはすでに評価結果を [7] 及び [6] において示してあるので結果のみを以下に示す。全て単位は sec であり、exec が CPU 処理時間、 trans が転送時間、total が 全処理時間を表し、括弧内の値は total に対する比率を表す。また、64 台の処理時間を 1 とした speedup のグラフを図 2 に示す。

転置方式による FT の総合結果

PU 数	exec	(%)	trans	(%)	total
64	5.52	(87.4)	0.79	(12.6)	6.31
256	1.36	(80.0)	0.34	(20.0)	1.71
1024	0.33	(68.0)	0.16	(32.0)	0.49

固定方式による FT の総合結果

PU 数	exec	(%)	trans	(%)	total
64	10.77	(96.1)	0.44	(3.9)	11.21
128	4.38	(94.4)	0.26	(5.6)	4.64
256	1.65	(91.8)	0.15	(8.2)	1.80
512	0.49	(85.5)	0.08	(14.5)	0.57
1024	0.24	(83.7)	0.05	(16.3)	0.29

IS の総合結果

PU 数	exec	(%)	trans	(%)	total
64	2.84	(99.4)	0.0177	(0.6)	2.86
128	1.36	(98.4)	0.0222	(1.6)	1.38
256	0.64	(95.4)	0.0308	(4.6)	0.67
512	0.29	(85.9)	0.0478	(14.1)	0.34
1024	0.14	(63.4)	0.0826	(37.6)	0.22

FT は データの mapping を動的に行なう転置方式 (FT-TRANS) と、mapping を固定的に行なう 固定方式 (FT-FIX) の 2 種類の並列化で評価を行ない、PU 台数を増やした時に固定方式では、superlinear 的な特性が得られた。また、IS は linear な特性を示している。

6 EP (完全並列問題)

EP はモンテカルロ・シミュレーションの典型例であり、ほぼ完全な並列化ができる。従ってこの結果、PU 単体の浮動小数点演算性能の実効的な上限を示すことになる。

具体的な処理は以下の計算を 2^{28}回行なうものである。。

1. 乱数生成

   48bit の乗算合同法

$$x_{i+1} = ax_i (\mathrm{mod}\ 2^{48})$$

を用いて、2 つの一様分布乱数 $x, y(-1 < x < 1, -1 < y < 1)$ を生成する。

2. Gaussian deviate の計算

   $t = x^2 + y^2 \leq 1$ の個数を数え上げる。また、Gaussian deviate

$$X = x\sqrt{(-2\log t)/t}$$

$$Y = y\sqrt{(-2\log t)/t}$$

   を計算し、$\max(|X|, |Y|)$ の小数点以下を切捨てた値の頻度表を作成する。

乗算合同法による乱数生成は基本的に逐次処理であるが、第 j 番目の乱数 x_j は $O(log_2 j)$ 回の演算で求めることができる [2]。従って、全部で 2^{28} 個の乱数組を P 台の PU で等分割し、各 PU は自分の担当の先頭乱数組をまず求める。以後各々の担当する部分列を独立に求め、処理を行ない、最後に頻度表を集計する。この時の PU 間通信は reduction 処理であり、binary-tree を用いて行なう。

転送時間を含めた全処理時間を表 3 に示す。また、64 台の処理時間を 1 とした speedup のグラフを図 2 に示す。

表 3: EP の総合結果

PU 数	exec	(%)	trans	(%)	total
64	16.135	(99.9)	0.000031	(-)	16.135
128	9.499	(99.9)	0.000037	(-)	9.499
256	4.066	(99.9)	0.000042	(-)	4.066
512	2.062	(99.9)	0.000047	(-)	2.062
1024	1.052	(99.9)	0.000052	(-)	1.052

EP の処理時間は、CPU 時間がほとんどを占める結果となり PU 台数を変化させた時にはほぼ完全な台数効果を示す。128PU の 時のみ台数効果が低いが、これはメインルーチン中のメモリ領域と数学ライブラリ中で使用するテーブルのメモリ領域のキャッシュ・ライン・コンフリクトが原因と考えられる。実験当初では 256 PU の場合でもこの現象が起きたが、メモリ領域のアドレスを変えた結果 linear な性能が出た。128 PU の場合についてもアドレスの変更を行なったが性能は改善されず、この点に関しては今後さらに詳しい解析を行なう必要があると考える。

7 MG (multi-grid 法)

MG は V-cycle multi-grid 法によるポアソン方程式の求解をおこなうプログラムである。格子点数 256×256 の u について、ポアソン方程式

$$\nabla^2 u = v$$

を周期境界条件の下で解く。v は全格子点中に $+1$ と -1 がそれぞれ 10 箇所あり、残りの点では 0 である。

MG は各点に対して、周囲 27 点のデータによる値の更新処理を繰り返す。この結果、各 PU は割り当てられた データに対する値の更新と隣接する PU 間でのデータ転送を繰り返すことになる。PU 空間は 1 次元、2 次元、3 次元が考えられるが、データ転送が近接格子転送のみなので、データの体積と表面積の関係から 3 次元 PU 空間への mapping がデータ転送量の面からは有利である。しかし、この mapping では 3 つの次元方向へのデータ転送必要なために転送パターンが複雑になる。これに対して、1 次元 PU 空間への mapping はデータ転送が簡単化されるが、データ転送量は多く、そもそも 1 次元方向の並列性が 256 PU 分 しかないので、今回の評価には不適当と考えられる。以上の考察から 2 次元 PU 空間を用いて行なう。

また、MG の元々の問題空間は 3 次元で 256^3 であるが、これは処理の途中で変化する。V-cycle multi-grid の途中では 問題サイズの大きさは $(1/2)^3$ ずつ小さくなっていき、最終的には 2^3 の大きさになる。そして今度は 2^3 倍ずつ大きくなり、最後に元の 256^3 の大きさになる。1024PU を 32×32 の 2 次元 PU 空間に展開して mapping をした場合、問題空間の大きさが 32^3 までは全 PU で行なう並列性があるが、これより小さくなった時は並列度が PU 数を下回る。この状態で無理に並列処理を進めても処理の粒度が小さく、かえって速度低下を招く危険がある。このため、このサイズより小さい処理は 1PU でまとめて行なう。

MG は working set size が大きいこと、データ値の更新と転送を交互に行うことにより、キャッシュが有効に働かない。これに対して、ここでの評価ではデータ値の更新には PVP-SW を用いたベクトル化を行なう。

MG 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、PU 台数が 1024, 256, 64 の場合についてのみ評価を行なった。全処理時間を表 4 に示す。また、64 台の処理時間を 1 とした speedup のグラフを図 2 に示す。

表 4: MG の総合結果

PU 数	exec	(%)	trans	(%)	total
64	1.143	(98.6)	0.016	(1.4)	1.159
256	0.375	(97.3)	0.010	(2.7)	0.386
1024	0.245	(96.4)	0.009	(3.6)	0.255

全処理時間に対して転送時間の占める割合は 1.4% から 3.6% であり、全処理時間に対する転送時間の影響は

少なく、全処理時間はCPU処理時間に依存していることが分かる。CPU処理時間は、PU台数が多い場合に逐次に処理を行なう部分が増えることが原因で、64 PUの時と256 PUの時の台数効果はほぼ同程度であるが、1024 PUの時には著しく低下する結果となる。この逐次処理の部分が、全処理時間に占める割合を概算してみる。各gridにおける演算量はその点数に比例すると考えて良いので、全処理時間は

$$T_1 = \sum_{i=1}^{8} 2^{3i}$$

のオーダになる。このうち、全PUで並列に実行できない部分は

$$T_{seq} = \sum_{i=1}^{4} 2^{3i}$$

であり、並列処理の部分と逐次処理の部分の比は

$$(T_1 - T_{seq})/1024 : T_{seq} = 1 : 0.250$$

となる。この割合は無視できるとはいい難く、PU台数が増加した場合に、全PUによる並列処理ができなくても、一部のPUを用いて並列処理を行なうように、処理のアルゴリズムを変更することが必要であろう。このような改良を行なった評価は今後の課題として挙げられる。

8 CG (CG法)

CGは正値対称な大規模疎行列の最小固有値をCG (Conjugate Gradient) 法により求めるプログラムである。ここで用いる大規模疎行列Aは正値対称であり乱数を用いて生成する。

CG法で行う演算は行列とベクトルの積、ベクトルの内積、ベクトルとスカラの積和計算であるが、行列とベクトルの積 ($q = Ap$) が最も重い処理なのでこの部分を中心に並列化を考え、以下の2つの方式を考える。

1. 行分割方式

   PUを1次元に展開し行列Aを行分割し、qもAと同様に行分割する。pは担当するAの列要素に対応する行要素だけがあればよいが、基本的に全要素が必要となる。CG法は反復法であり、pはqを元にして作られるので、各PU内で求めたqの一部を全PUに転送する必要がある。この転送は全対全broadcastとなり、PU数Pに対して$O(P)$の転送回数がかかるので、PU台数が多い時にボトルネックになる可能性がある。

   内積計算はベクトルの担当行分の内積を計算し、reductionとbroadcastを用いることで並列化する。また、ベクトルとスカラの積和計算は単に担当行毎の計算を行なう。

2. 列分割方式

   PUを1次元に展開し行列Aを列分割する。この場合、pはAの列に対応する要素をPUで分割して持つ。しかし、各PUで計算したqはreductionとbroadcastを行なう。転送回数はPU台数Pに対して$O(logP)$であるが、転送量は大きい。

   ベクトルの内積計算や、ベクトルとスカラの積和計算はベクトルを行で分割し、行分割方式と同様に行なう。

最初にCGの転送時間の結果を表5に示す。単位はsecであり、rowが行分割方式の転送時間、columnが列分割方式の転送時間を表す。

表5: CGの転送時間の結果

PU数	row	column
64	0.610818	2.618505
128	0.828465	2.992559
256	1.252277	3.366613
512	2.090961	3.740667
1024	3.762119	4.114721

行分割方式は全対全のブロードキャスト転送を行なうのでPU数Pに対してデータ転送回数が$O(P)$で増加する。これに対して、列分割方式では$O(logP)$の転送回数で行なうことができる。しかし、列分割方式は、一回の転送量がPU数に関係なくベクトルの14000要素全てであり、メッセージ長が$N\frac{1}{2}$より大きく転送効率は良いが、データ転送時間がかかる。実験当初、この二つの方式は転送時間の増加の傾向に違いがあるので、PU台数によっては転送時間が逆転する関係があると予想したが、実験の結果から、表5に示すように行分割方式の方が列分割方式よりも常に短い時間で転送できる結果となり、転送時間の逆転は存在しないことが分かった。

行分割方式の全処理時間を表6に示す。また、64台の処理時間を1としたspeedupのグラフを図2に示す。表6からCPU時間がPU台数Pに対して、$O(1/P)$で減少していることから、CPUの処理はほぼ並列化されていることが分かる。列分割方式のCPU時間の測定は行なっていないが、ベクトルqのreductionの際に14000要素に対する加算が必要となり、行分割方式よりも長いCPU時間がかかると考えられる。さらに、表6から、全処理時間に対して転送時間が大きな割合を占めていることが分かる。列分割方式を用いることでPU数が多い時に転送時間を短くし、全処理時間を

表 6: CG の総合結果

PU 数	exec	(%)	trans	(%)	total
64	1.433	(70.1)	0.611	(29.9)	2.044
128	0.781	(48.5)	0.828	(51.5)	1.610
256	0.430	(25.6)	1.252	(74.4)	1.682
512	0.266	(11.3)	2.091	(88.7)	2.357
1024	0.185	(4.7)	3.762	(95.3)	3.947

短縮できると考えたが、その転送量の多さから転送時間は短縮できず、全処理時間の短縮も見込めない。

行分割方式と列分割方式を組合せ、疎行列 A を メッシュ状に分割し 2 次元 PU 空間に block 分割 mapping する方式も考えられ、これらの問題に対して、最適な分割方式と問題規模、PU 台数の関係を求めることは今後の課題である。

9 他の並列計算機との比較

他の並列計算機における NASPB の結果 [1] と、CP-PACS の評価結果を比較する。他の並列計算機については 64PU のデータが公表されているので 64PU での CP-PACS 及び 他の並列計算機の処理時間を図 3 に示す。各々の並列計算機において、各問題をどのように並列処理しているかは明らかではないが、CP-PACS と同様にデータ・パラレルの並列化を行なっていると考えられる。

EP では、PU 単体の性能差が現れており、CP-PACS は他の並列計算機に対してそれほど高速ではないことが分かる。これに対して、他の問題に関しては、高速な部類に属しているので、実アプリケーションを並列処理した時の性能は、その PU 単体の性能差を考慮すると、CP-PACS は良い結果を示すといえる。今回の評価で EP がそれほど高速でない理由は以下のように考えられる。一つは想定したプロセッサの理論的な peak 性能は、1 クロックで加算と乗算の浮動小数点演算を行なった場合に得られるが、これは実際の問題で常に有効であるとは限らない。このような複合命令を除くと T3D の DEC Alpha や、SP2 の POWER2 に対して、プロセッサが比較的低速であるといえる。これに対しては、動作クロックを上げるなどの対策が考えられ、実際の CP-PACS の計画では 100MHz よりも高い動作クロックを目指している。もう一つの理由としては、コンパイラやライブラリがプロセッサの superscalar を十分に意識して作られてはいないことが挙げられる。これに対しても、CP-PACS では独自のコンパイラ、ライブラリを開発することで、高速化を目指す予定である。

MG での転送は近接格子転送であり、どの並列計算機においてもネットワークトポロジに依存するデータ転送時間の差は生じないと考えられる。この問題で CP-PACS が高い性能を示している要因は、CPU 処理を PVP-SW 化したことにより、プロセッサ本来の性能を十分に引き出し、高速に処理を行なうことができた点にあると考えられる。MG は PU 台数が多い時には並列化が十分でないと先の考察で述べたが、64PU ではこれは問題にならない。この結果 CP-PACS の結果は他の並列計算機に比べて良い結果を示しているといえる。

64PU の CP-PACS で CG を行なった結果は、全処理時間に対してデータ転送時間が 29.9% という高い比率を示しており、他の並列計算機においてもデータ転送の割合が高いことが予想される。これは転送パターンが全対全 broadcast なので、PU 台数のオーダの転送回数が必要となり、1 回のメッセージ転送の overhead の累積が問題となる。これに対して、CP-PACS では PU 間 DMA 転送によって overhead を小さくしたため、良い結果が示されていると考えられる。

CG を除いて、CPU 時間が全処理時間のボトルネックとなっているので、相互結合網は想定した性能のままで、動作 clock を上げるなどの PU 単体の性能向上は全処理時間の短縮に役立つと考えられる。

10 結論

本研究では NASPB による CP-PACS の仮想性能評価を行なった。NASPB のカーネルの問題に対して、CP-PACS に適した並列プログラムを実際に作成し、それを基に転送時間と CPU 時間を評価した。転送時間は転送パターン、転送回数、転送量、ネットワークの諸性能から、机上計算によって求め、CPU 時間は プロセッサ用のシミュレータを用いてシミュレートし、実行クロックサイクル数を計測することにより評価した。プログラムは主に C 言語によって記述し、PVP-SW による擬似ベクトル化を行なう部分については、アセンブラによって記述した。PVP-SW によるベクトル化は、PU 間のデータ転送によってキャッシュを有効に用いることができない部分について行なった。

全体として、CP-PACS は 相互結合網に HXB を用いたことにより、各種の問題に対して、データ空間を PU 空間へマッピングする方法が自由に選べ、なおかつ、いずれの場合でもデータ転送を無衝突で行なうことができた。これによって、ネットワークのトポロジを意識することなく、問題の並列化を行なうことができた。また、CPU 処理に関しては、PVP-SW を用いることで、キャッシュの制約を受けることなしに転送をともなうデータに対しても高速に処理することができることを明らかにした。MG と CG の問題に関しては、

PU 台数を増やした時に台数効果が出ない結果となったが、これらについては、並列化の手法に改良の余地があり、再評価をする必要がある。

今後の課題は、並列化に改善の余地のある問題に関して再評価を行なうこと、転送と内部処理のオーバラップを行なう処理方式により評価を行なうこと、PVP-SW を考慮したコンパイラ・ライブラリを用いて評価を行なうことが挙げられる。

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, Eric Barszcz, Leonardo Dagum, and Horst D. Simon. "THE NAS PARALLEL BENCHMARK RESULTS 10-94". Technical report, NAS, October 1994. RNR-94-001.

[2] D. Bailey, E. Barszcz, J. Barton, D. Browning, R. Carter, R. Fatoohi, P. Frederickson, T. Lasinski, R. Schreiber, and H. Simon. "THE NAS PARALLEL BENCHMARKS". Technical report, NAS, March 1994. RNR-94-007.

[3] 朴泰祐, 斉藤哲也, 板倉憲一, 中澤喜三郎, 中村宏. 「ハイパクロスバ・ネットワークの性能評価」. 信学技報 CPSY93, November 1993.

[4] Al Geist, Adam Beguelin, Jack Dongara, Weicheng Jiang, Robert Manchek, and Vaidy Sunderam. *"PVM3 USER'S GUIDE AND PEFERENCE MANUAL"*, May 1993.

[5] 位守弘充, 中村宏, 朴泰祐, 中澤喜三郎. 「スライドウィンドウ方式による擬似ベクトルプロセッサ」. 情報処理学会論文誌, Vol. 34, No. 12, pp. 2612-2613, December 1993.

[6] 板倉憲一, 廣野哲, 朴泰祐, 中村宏, 中澤喜三郎. 「PVP-SW とハイパクロスバ・ネットワークを用いた計算機の評価」. 情処全大 第 49 回 (6), pp. 55-56, September 1994.

[7] 板倉憲一, 廣野哲, 朴泰祐, 中村宏, 中澤喜三郎. 「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」. 情処研報 HPC-52-20, pp. 119-126, July 1994.

[8] H.Nakamura, H.Imori, K.Nakazawa, T.Boku, I.Nakata, Y.Yamashita, H.Wada, and Y.Inagami. "A Scalar Architecture for Pseudo Vector Processing based on Slide-Windowed Registers". In *Proc. of ACM International Conference on Supercomputing '93*, pp. 298-307, July 1993.

[9] 中澤喜三郎, 朴泰祐, 中村宏, 中田育男, 山下義行, 岩崎洋一. 「CP-PACS のアーキテクチャの概要」. 情処研報 ARC-108-9, October 1994.

[10] 田中輝雄, 面田耕一郎. 「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, June 1990.

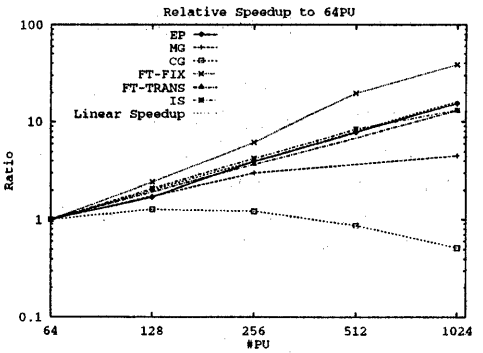

図 2: 全処理時間と PU 台数の関係

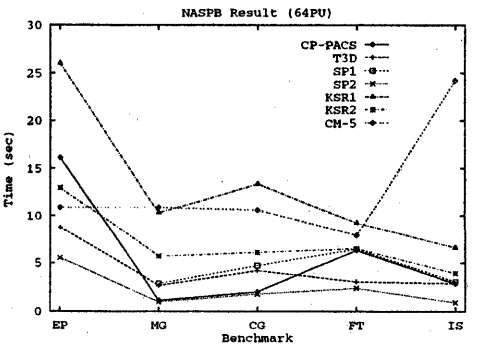

図 3: 64PU における各ベンチマーク問題の処理時間