計算機アーキテクチャ 123－4
（1997. 3. 7）

超並列計算機 CP-PACS の基本性能評価

板倉憲一[†] 安部井嘉人[†] 松原正純[†]
朴泰祐[†] 中村宏[††] 中澤喜三郎[†††]

本研究ではさまざまな並列数値処理プログラムで用いられるカーネルループや基本転送パターンによって，超並列計算機 CP-PACS の単体プロッセッサの処理能力とプロセッサ間相互結合網の性能を実機によって測定し，その解析を行った．結果から擬似ベクトル処理機構及び 3 次元 Hyper-Corssbar ネットワークの性能が，シミュレーションによる評価と同様に高い性能を持つことが分った．これらの性能により，CP-PACS が各種大規模科学技術計算に対し非常に高い潜在能力を持つことが確認された．

Evaluation of the basic performance of CP-PACS

KEN'ICHI ITAKURA,[†] YOSHITO ABEI,[†]
MASAZUMI MATSUBARA,[†] TAISUKE BOKU,[†]
HIROSHI NAKAMURA[††] and KISABURO NAKAZAWA [†††]

The massively parallel processor CP-PACS has two special features', PVP-SW (Pseudo Vector Processor based on Slide Window) for tolerating memory access latency and 3 dimentional Hyper-Crossbar Network which provides very high bisection bandwidth.

In this research, we evaluate the CPU performace from kernel loops and the network performance from basic data transfer patterns on CP-PACS. As an analysis of these results, it is shown that the potential of CP-PACS is very high and suitable for various large scale scientific problems.

1. はじめに

筑波大学では計算物理学を主目的とした CP-PACS プロジェクト[1]が進められており，大規模科学技術処理用の超並列計算機 CP-PACS[2)3)]を開発した．CP-PACS は本来の目的である QCD 計算の高速化を主眼としているが，その他の様々な数値処理問題に対しても有効に働くようにそのアーキテクチャが設計されている．特に，従来のスーパコンピュータでさえも能力不足となるような大規模問題に対して，数千台規模という超並列の特性を活かすための強力な相互結合網を用意し，また，単体プロセッサの処理能力の向上を求めて新たなアーキテクチャを導入している．

本報告では，この CP-PACS の単体プロセッサの特徴である，擬似ベクトル処理機構 (PVP-SW: Pseudo Vector Processor based on Slide Window)[4)] と，プロセッサ間相互結合網である 3 次元 Hyper-Crossbar 網 (HXB)[5)] についての基本性能評価を行い，その有効性について考察する．

2. 超並列計算機 CP-PACS

CP-PACS[2)3)] は 2048 台の PU (Processing Unit) を疎結合した分散メモリ型超並列計算機であり，その理論最高性能は 614.4 GFLOPS である． CP-PACS の構成図を図 1 に示す．

CP-PACS の PU はプロセッサ，主記憶，キャッシュメモリ (1 次データ: 16KB，2 次データ: 512KB，いずれもダイレクトマップ)，ネットワークインターフェイスアダプタ (NIA)，記憶制御装置 (Storage Control Unit) から構成される．CP-PACS は MIMD 動作方式を用い，ユーザプログラム中で明示的に記述されるメッセージパッシング命令によってデータを交換する．以下に CP-PACS の主な特徴である，プロセッサの PVP-SW 機構と PU 間相互結合網 HXB について述

† 筑波大学 電子・情報工学系
Institute of Imformation Sciences and Electronics, University of Tsukuba
†† 東京大学 先端科学技術研究センター
Research Center for Advanced Science and Technology, University of Tokyo
††† 電気通信大学 情報工学科
Department of Computer Science, University of Electro-Communications

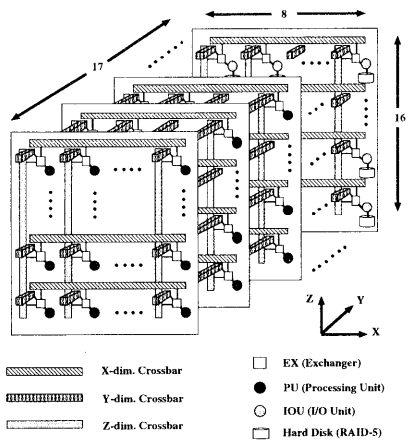

図1　CP-PACS の構成図

べる．

PVP-SW[4] 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 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する．その際，既存 RISC プロセッサとの命令アーキテクチャレベルの上位互換性を保つため，拡張したレジスタにはレジスタ・ウィンドウを用いてアクセスし，通常の命令に対しては拡張前のレジスタ数と同等になるようにする．preload 命令などの拡張した命令は全てのレジスタにアクセス可能にし，実際の演算に先行して preload 命令を発行可能にする．

擬似ベクトル処理はスカラプロセッサのループ処理として実現し，ループの 1 イタレーション内では実際の演算の他に，後の演算で対象となるレジスタへの preload と，ウィンドウを前方にスライドさせる命令を含む．また，スーパスカラ方式により preload 命令と演算命令の同時処理を行うことにより，ベクトル処理を効率的に行う．

CP-PACS の PU では，そのメモリシステムの制約から store および poststore を連続的に行うときには 2 サイクルピッチでしか処理ができない．これは load/store ネックのプログラムではボトルネックとなりやすく，この点については評価を行うときに考察する．また，擬似ベクトル処理はキャッシュと併用することが可能で，CPU をストールさせないソフトウェアによるキャッシュ・プリフェッチ (prefetch) も可能である．メモリはパイプライン・アクセスが可能なように 16 バンク構成となっている．

次に，PU 間相互結合網について述べる．図1は CP-PACS の 8 × 17 × 16 の 3 次元 HXB を示している．この図のように，PU は 3 次元直交空間上に並べられ，各次元方向にそって一列に並んだ PU 同士はクロスバスイッチ (XB) によって完全結合される．3 次元 HXB の場合，1PU は 3 方向の XB と接続され，そのポイントにエクスチェンジャ(EX) と呼ばれるルータスイッチが設けられている．HXB の特徴は中規模のクロスバで接続されているのでその半径が小さく，また，ネットワークのリンクの数が非常に多いのでそのバイセクションバンド幅が広い点である．

さらに，CP-PACS はリモート DMA と呼ばれる高速なデータ転送方式を用いている．これは，PU の NIA がユーザメモリ空間のデータを直接 DMA アクセスして送受信を行うものである．特に受信側では受信命令を発行する必要は無く，メッセージ到着によって自動的に NIA が動作し，受信側のユーザメモリ空間にデータが書き込まれる．このように，システム領域のバッファを使用しないので，ハードウェアの高速なスループットをそのまま活かすことができる．また，転送の立ち上げオーバヘッドを軽減するために，NIA の起動は 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．

現在の CP-PACS は 2048 PU を 1024PU，512PU，256PU × 2 の 4 つのパーティションに分けて分割運転しており，今回の性能測定には 256 PU (4 × 8 × 8) のパーティションを使用した．

3. CPU 単体性能評価

CPU の性能は表1に示すカーネルループによって評価する．ここに示したループは全てコンパイラによって PVP-SW 機構による擬似ベクトル処理コードが生成可能であり，このコード (以下 PVPcode) と通常の load/store 命令を用いキャッシュメモリによってメモリレイテンシを隠蔽するコード (以下 CACHEcode) の比較を行う．以下では L をループ長とする．

表1　CPU 性能評価を行うカーネルループ

ループ名	内容
ADD	$c(i) = a(i) + b(i)$
DAXPY	$c(i) = s \times a(i) + c(i)$
SUMUP	$s = s + a(i)$
INNERP	$s = a(i) \times b(i)$
L_SUMUP	$s = s + a(l(i))$
L_INNERP	$s = s + a(l(i)) \times b(i)$

ベクトル a, b, c およびスカラ s は倍精度浮動小数点データであり，ベクトルはいずれも 2 次キャッシュより大きい容量を持つ．このベクトルの先頭から L 要素のデータを用いてループ長と処理時間の関係を測定する．なお，各ベクトルの先頭番地は 1 次および 2 次キャッシュメモリでラインコンフリクトしないように意図的

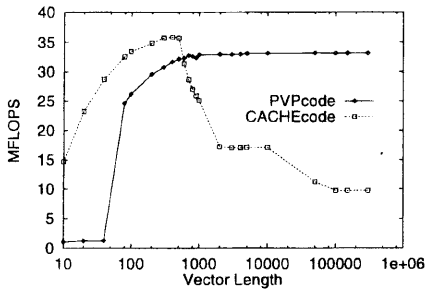

図2　ADD の実効性能

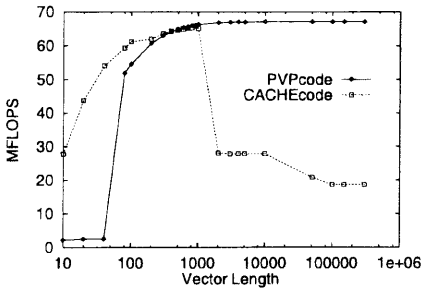

図3　DAXPY の実効性能

にずらして配置してある．また，リストベクトル処理 (L_SUMUP, L_INNERP) ではインデックスベクトル l が重複無しに a の10% の要素をランダムに指す．CACHEcode に対しては全てのデータがキャッシングの対象であるが，PVPcode ではインデックスベクトル l のみがキャッシングされる．これは，浮動小数点データは preload/poststore によってメモリアクセスレイテンシの隠蔽が可能であり，他のキャッシングすべきデータ (インデックスベクトル l) に干渉させないためである．このようなキャッシングのコントロールは FORTRAN ソース上のディレクティブによって宣言できる．一般のアプリケーションではここで測定するような基本ベクトル処理を繰り返し行うので，測定は複数回繰り返した場合で行い，キャッシュメモリが warm start の状態で評価を行っている．後に示す評価ではベクトル長が非常に短い時に PVPcode が低い性能しか出ていないが，これはコンパイラがループを擬似ベクトル処理するのが不可能と判断し，キャッシングしないデータに対しても通常の load/store 命令によってアクセスするコードを使用するからである．

まず，ベクトルの store のある ADD と DAXPY の結果を図2，3 にそれぞれ示す．CACHEcode の ADD と DAXPY では $L < 500$ の時にワーキングセットが1 次キャシュメモリ内に収まり，$L < 10000$ の時に2 次キャシュメモリ内に収まる．図2，3ではループ長が長くなるとキャッシュメモリが容量不足となり，性能低下が顕著に現れる．これに対して，PVPcode では長いベクトル長に対してもメモリアクセスレイテンシを十分隠蔽し，高い性能を維持できることが分かる．コンパイラは ADD の FORTRAN ソースを CACHEcode，PVPcode 共に 4 回アンローリングする．load/store 命令と演算命令はスーパスカラによって 同時発行可能なため，ADD は load/store ネックのループとなる．このループの処理の実効演算性能 S[MFLOPS] は 1 イタレーションの演算数 f[FLOP] とその実行クロック数 t[clock] より以下のように計算できる．

$$S = f \times 150/t$$

そこで，CACHEcode と PVPcode の最内ループのループ演算におけるアセンブラソースを解析し，スーパスカラ性を考慮してコードスケジュールを検討した．このときに CACHEcode では全データがキャッシュヒット，PVPcode では preload/postsotre によってメモリアクセスレイテンシが完全に隠蔽できていると仮定した．このようにして，ADD 以下全てのカーネルについて予測した性能を表2に示す．

表2　各カーネルのピーク性能予測 [MFLOPS]

ループ名	CACHE		PVP		NOPF	
ADD	35.3	(17)	33.3	(18)	-	
DAXPY	66.6	(18)	66.6	(18)	-	
SUMUP	120.0	(5)	100.0	(6)	-	
INNERP	133.3	(9)	120.0	(10)	-	
L_SUMUP	66.7	(9)	54.5	(11)	60.0	(10)
L_INNERP	100.0	(12)	80.0	(15)	85.7	(14)

(括弧内はスケジューリングを考慮した実効クロック数である．表中 CACHE，PVP，NOPF はそれぞれ，本文中の CACHEcode, PVPcode, NOPFcode に対応する．)

PVPcode ではウィンドウ切り替えによって，ループオーバヘッドが大きいために，ピーク性能予測では CACHEcode の方が高く，実際にもワーキングセットが 1 次キャッシュに収まる短ベクトルでは warm start の CACHEcode の方が勝っている．しかし，この差はアンローリングの度数を増やすことで縮まるので，PVPcode はさらなるアンローリングをすることが必要だと考える．DAXPY においてもほぼ同様な load/store ネックのスケジューリングとなるが，CACHEcode に整数系演算が 1 つ入るために，理論的な性能が等しくなっている．なお，減算や乗算のループも同じコードスケジュールになるので ADD と同様の結果となる．

次に，2 サイクルピッチの store の影響を除いた例として，SUMUP と INNERP の結果を図4，5 にそれぞれ示す．SUMUP と INNERP は，ADD と同じく 4 回アンローリングされているが，ループ内に store が無いので，4 load もしくは 8 load 分の時間とループオーバヘッドで理論ピーク性能が計算できる．

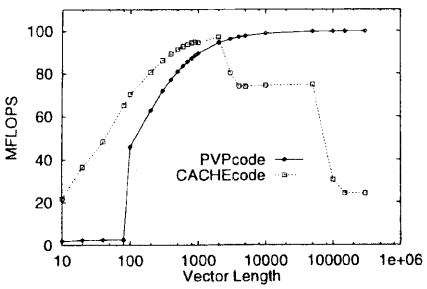

図4 SUMUP の実効性能

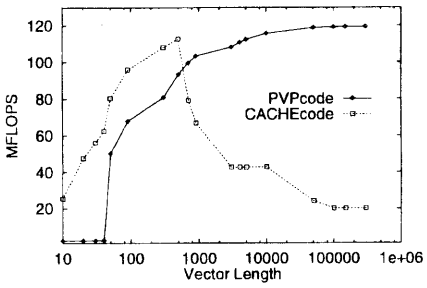

図5 INNERP の実効性能

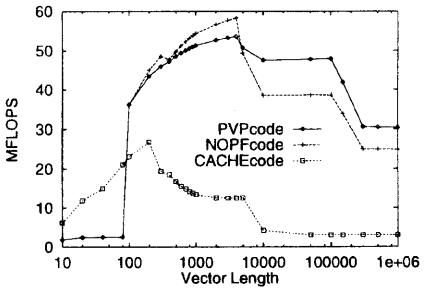

図6 L_SUMUP の実効性能

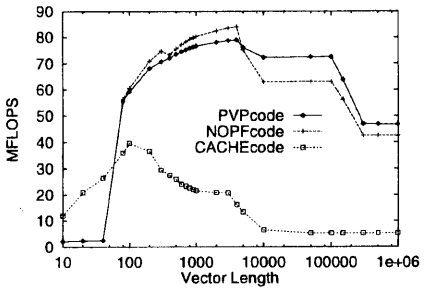

図7 L_INNERP の実効性能

この2つのループでは ADD に比べて最内ループが短いので，ループの立ち上げオーバヘッドの影響が大きい．CACHEcode は 理論ピークに到達する点が1次キャッシュメモリの容量とほぼ等しく，それ以下のベクトル長では性能が発揮できていない．これに対して，PVP-SW では同程度のベクトル長で，ほぼピーク性能に達し，それ以上のベクトル長でも速度低下が無いことが分る．PVPcode の測定結果は表2の解析結果に一致するが，CACHEcode は理論ピークに達する前に1次キャッシュメモリの容量不足を起している．

最後に疎行列演算などで表われるリストベクトル処理の例として L_SUMUP，L_INNERP の結果を図6，7 に示す．CACHEcode では浮動小数点データとリストインデックスデータをキャッシングするので，ベクトル長が高々100 程度で性能低下を引き起す．このような短いベクトル長ではループの立ち上げオーバヘッドの影響が大きく warm start の状態でも高い性能が出ない．キャッシュメモリに頼ったシステムでは，大きなワーキングセットサイズをいくつかのブロックに分割し，高いキャッシュヒット率の下で演算を行う高速化手法が一般に行われる[6]．しかし，リストベクトル処理のようにワーキングセットに対して演算数が少ない例では，ループ分割によるオーバヘッドが大きく，高い性能を発揮することが難しいことをこの例は示している．これに対して，PVPcode では高いメモリスループットを活かし大きなワーキングセットに対しても有効に働く．PVPcode ではリストベクトルデータ $a(l(i))$ は以下の手順でレジスタに読み込む[7]．

(1) $l(i)$ をキャッシュメモリへ prefetch
(2) $l(i)$ の load
(3) $a(l(i))$ の preload

しかし，$L < 4000$ では l を全てキャッシュメモリに保持できるために prefetch は余分な時間消費である．そこで，最内ループ内で prefetch を行わないコード (NOPFcode) を作成し，その測定結果も図6，7 に示してある．確かに $L < 4000$ では NOPFcode の方が PVPcode よりも良い性能を出しているが，これは，ループアンローリングの割合いを増やすことで差が縮まる．しかし，$L > 4000$ の場合には NOPFcode は l のメモリアクセス時間を隠すことが難しくなるので，その性能低下の割り合いは大きい．$L > 140000$ では l が2次キャッシュにも収まらなくなる．この時には，PVPcode のキャッシュプリフェッチを行っても l のメモリアクセス時間を隠すことができなくなり，共にピークの半分程度の性能となってしまう．このような場合にはそもそも十分なベクトル長があるので，それを活かして十分手前にキャッシュプリフェッチ命令を発行しておき，2次キャッシュに収まっている程度の性能を保持できるのではないかと考えられる．

4. Network 性能評価

Network の性能評価はバリア同期と表3に示すデータ転送パターンで行う．バリア同期はシステム関数と

して用意されており，内部では参加する PU によって4進木を構成し，ソフトウェアによってバリアメッセージの収集および放送を行う．連続する2回のバリア同期の間の時間と PU 数の関係を表4に示す．この結果よりバリア同期の時間は log_4(PU 数) に比例した時間がかかっていることが分り，256 PU での同期にかかる時間は約 80 μsec である．

表3　Network 性能評価をデータ転送パターン

転送パターン名	内容
PINGPONG	隣接 2PU によるピンポン転送
PARALLEL	等方一斉転送
	[dst=(src+1)%P]
SHUFFLE	シャッフル転送
	[dst=(src<<1)%P
	+(src>>(log_2(P)-1))]
RANDOM	ランダム転送
ALL2ALL_BT	全対全ブロードキャスト (バタフライ転送)
ONE2ALL_SYS	1 対全ブロードキャスト (システム関数)
ONE2ALL_PP	1 体全ブロードキャスト (P-1 回転送)
ONE2ALL_2TR	1 対全ブロードキャスト (2 進木による転送)

(PARALLEL と SHUFFLE は 全PU に1次元アドレスを振り，送信元 PU (src) と送信先 PU (dst) の関係を C 言語の書式で示した．P は PU 台数を表す．)

表4　バリア同期の時間

PU 数	16	32	64	128	256
時間 (μsec)	38.8	48.3	61.4	69.5	80.9

まず，1PU からメッセージを1回転送するのにかかるレイテンシを PINGPONG，PARALLEL，SHUFFLE の転送パターンにおいて測定した結果を図8に示す．ここで，PINGPONG は 往復の時間の半分で測定しており，そのためにほとんど誤差がないと考えられるが，PARALLEL，SHUFFLE の計測はまず，全 PU でバリア同期を取り，その後転送が終了するまでの時間を各 PU が測定し，その中で最も時間のかかったものを示している．しかし，256 PU でのバリア同期の時間が 80 μsec であることから，メッセージ長が 8KB よりも短い場合にはこのバリア同期の誤差にまぎれてしまい，測定することが不可能であった．以下では，8KB よりも大きいメッセージについての考察を行う．

PINGPONG は最も基本的な転送パターンであり，その転送性能はトポロジに依存しないハードウェアの基本性能と言える．PARALLEL は仮想的な1次元アドレスの隣りに送る転送であり，実際の3次元 HXB では必ずしも隣りに送るわけではない．しかし，HXB ではこのような等方転送は完全に無衝突に行えるので[8]，PINGPONG とほぼ同じレンテンシとなる．SHUFFLE 転送では3次元 HXB で最大2回衝突することがわかっている[9]．このために，PINGPONG

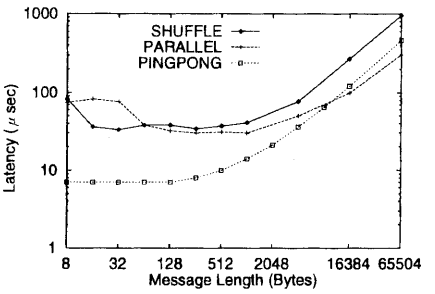

図8　各種転送パターンにおけるレイテンシ (256PU)

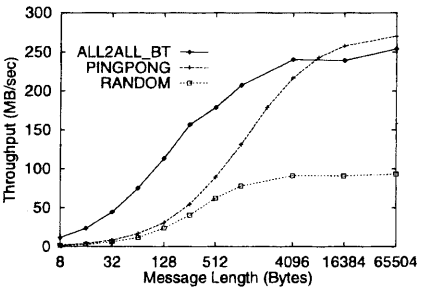

図9　各種転送パターンにおける実効スループット (256PU)

の3倍の時間がかかる．この2回の衝突は3次元という構成に起因しており，PU 数には直接的な関係がない．しかしこれはトーラスネットワークにおいてグローバルな通信をした場合の衝突の回数に比べると，非常に低い割合いである．

HXB の実効スループット性能を PINGPONG，RANDOM，ALL2ALL_BT の転送パターンにおいて測定した結果を図9に示す．この場合も PINGPONG の性能が指標となる．RANDOM では各 PU が乱数によって送信先を決定し，メッセージを送出することを繰り返す．この結果，各 PU は常にネットワークにメッセージを転送しようとし，非常に高い負荷状態の性能を調べることになる．なお，リモート DMA 機構によって NIA が受信メッセージを直接メモリに書き込むため，メッセージの受信操作は行わない．

メモリスループットおよび，NIA のスループットは送受信を同時に行うバンド幅を持つため，RANDOM の場合のスループットの低下は3次元 HXB 上でのメッセージの衝突によるものである．我々が過去に行った各種ネットワークトポロジのシミュレーション結果[5]では，他のトポロジに比べて3次元 HXB におけるランダム転送無衝突時の約 33 %と，他のネットワークに比べ非常に高い結果を示している．このシミュレーションではデータ転送時の立ち上げオーバヘッドが無い状態を仮定しており，今回の実例において，$N_{\frac{1}{2}}$ = 2KB よりもメッセージ長が大きい場合にはほぼシミュ

レーション通りの性能を示している．$N_{\frac{1}{2}}$ よりも短いメッセージ長では，PU 内でのソフトウェアおよびハードウェアのオーバヘッドの割合いが大きくなり，ネットワークに対する負荷が小さくなる．このためにネットワーク中の衝突が少なくなり無衝突の PINGPONG に近いスループットが得られる結果となった．

ALL2ALL_BT ではバラフライ転送パターンで全対全ブロードキャストを行っており[10]，横軸のメッセージ長は各 PU が最初に出すメッセージ長である．実際のメッセージ長はバタフライ転送の段数が進む毎に 2 倍になるので，平均のメッセージ長は長くなり，見かけのスループットが PINGPONG を越える．この割り合いは最初のメッセージ長が短い場合に顕著に現れ，最初の転送のメッセージ長が $N_{\frac{1}{2}}$ を越える場合のスループットはほぼ PINGPONG に等しくなり，最後には逆に段数分の転送オーバヘッドによって PINGPONG を下回る値となる．

最後に，1 対全のブロードキャストを 3 種類の方法で行った結果を図 10に示す．ONE2ALL_PP ではブロードキャストする PU が残りの全 PU に個別に転送を行う．この為に，図 8 の PINGPONG の約 256 倍の結果となる．ONE2ALL_2TR では 256 台の PU で 2 進木を構成し，順次転送をユーザレベルで行うものある．これに対して，ONE2ALL_SYS はシステム関数として用意されているブロードキャストである．この内部では，同一 XB に結合された PU への転送は NIA のハードウェアによって連続的に行われるため，オーバヘッドが少ない．このために，メッセージ長が短い場合には，ONE2ALL_2TR よりも ONE2ALL_SYS が短時間で終了する．しかし，メッセージ長が長い場合には，OS の処理が途中に入らないので，ユーザプログラムで書いた ONE2ALL_2TR の方が短時間で終了する．

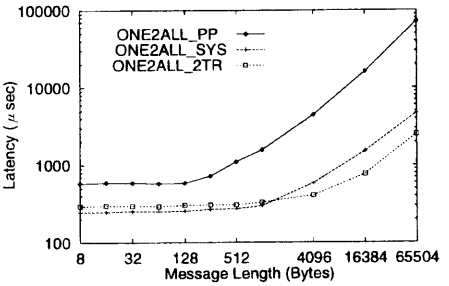

図10　1 対全ブロードキャストのレイテンシ (256PU)

5. おわりに

本報告では，超並列計算機 CP-PACS の基本性能をノードプロセッサとネットワークの性能に分けて測定，解析を行った．今までのシミュレーションや机上計算による性能予測では様々な点で理想的な状態を仮定した評価を行っていたが，実機においてもほぼ予想通りの結果が得られ，ノードプロセッサに用いた擬似ベクトル処理機構やノード間結合網である 3 次元 HXB が高い性能を持つことが分った．我々は，このような基本要素の性能解析は実アプリケーションのチューニングや，更に次期世代のアーキテクチャ開発に役立つものと考えており，このような基本性能をふまえて実アプリケーションのチューニング手法を提案することを考えたい．

謝辞　CP-PACS を利用する機会を与えて頂いた計算物理学研究センター関係各位に感謝します．また，筑波大学坂井修一助教授ならびにアーキテクチャ研究室の諸氏には貴重な御意見を頂いたことに感謝します．なお，本研究の一部は創成的基礎研究費 (08NP0401) の補助によるものである．

参考文献

1) 岩崎洋一ほか:「専用並列計算機による「場の物理」の研究」，技術報告，筑波大学計算物理学研究センター (1994)．新プログラムによる研究 研究進捗状況報告書．
2) 中澤喜三郎ほか:「CP-PACS のアーキテクチャの概要」，情処研報 ARC-108-9 (1994)．
3) Iwasaki, Y.: "Status of the CP-PACS Project", *Proc. of Lattice '96, Nucl. Phys. B(Proc. Suppl.)* (1996).
4) Nakamura, H. et al.: "A Scalar Architecture for Pseudo Vector Processing based on Slide-Windowed Registers", *Proc. of ACM International Conference on Supercomputing '93*, pp. 298–307 (1993).
5) 朴泰祐ほか:「ハイパクロスバ・ネットワークの性能評価」，信学技報 CPSY-93-40 (1993)．
6) Lam, M. et al.: "The cache performance and optimization of blocked algorithms", *Proc. of ASPLOS-IV* (1991).
7) Nakamura, H. et al.: "Pseudo Vector Processor for High-Speed List Vector Computation with Hiding Memory Access Latency", *Proc. of 1994 IEEE TENCON*, pp. 338–342 (1994).
8) 保田淑子ほか:「ハイパクロバネットワークの通信性能評価」，信学技報 CPSY-93-25 (1993)．
9) 板倉憲一ほか:「ハイパークロスバ・ネットワークにおける並列ソート処理」，情処全大 第 47 回 (6), pp. 177–178 (1993)．
10) 板倉憲一ほか:「並列計算機 CP-PACS の CG ベンチマークによる性能評価システム」，情処研報 HPC-63-6, pp. 31–36 (1996)．