計算機アーキテクチャ 108－9
（1994．10．28）

CP-PACSのアーキテクチャの概要

中澤 喜三郎†, 朴 泰祐†, 中村 宏†, 中田 育男†, 山下 義行†, 岩崎 洋一‡

筑波大学 電子・情報工学系†
筑波大学 物理学系‡
〒305 つくば市天王台1-1-1

Tel: 0298-53-6912 Fax: 0298-53-5206
{nakazawa,taisuke,nakamura,nakata,yaman,iwasaki}@rccp.tsukuba.ac.jp

概要 CP-PACSは，QCD計算に代表されるような計算物理学を中心とする科学技術計算を，超高速処理することを目的とした超並列計算機である．本報告では現在詳細設計を行っているCP-PACSのアーキテクチャの概要について述べる．

CP-PACSに採用されるノード・プロセッサは，汎用のRISCプロセッサに対しPVP-SW (Pseudo Vector Processor based on Slide-Windowed registers) と呼ばれるソフトウェア制御に基くベクトル処理機構を追加したもので，大規模科学技術計算において問題となるキャッシュ容量不足による性能低下を解決する．また，プロセッサ間結合ネットワークには高い交信性能と様々なデータ転送パターンに柔軟に対応するハイパクロスバ・ネットワークを用い，高速転送モードを始めとする特殊機構が提供されている．また，PVP-SWとシステム全体を効率的に動作させるためのコンパイラを中心とするソフトウェアも整備されつつある．

The Architecture of CP-PACS

Kisaburo NAKAZAWA† Taisuke BOKU† Hiroshi NAKAMURA†
Ikuo NAKATA† Yoshiyuki YAMASHITA† Yoichi IWASAKI‡

Institute of Information Sciences and Electronics, University of Tsukuba†
Institute for Theoretical Physics, University of Tsukuba‡
1-1-1 Tennodai, Tsukuba 305

Abstract CP-PACS is a massively parallel processing system for high-speed processing of scientific calculations represented by QCD calculation and other computational physics problems. In this paper, we describe the brief architecture of CP-PACS under designing now.

The node processor for CP-PACS is a general purpose RISC processor enhanced with a new feature named PVP-SW (Pseudo Vector Processor based on Slide-Windowed registers), which is a software controlled vector processing technique to avoid the performance degradation caused by the cache shortage problem. The Hyper-crossbar Network is used as the inter-PU communication network, which provides very high communication performance and flexibility for many kinds of data communication patterns. Some specialized features like high-speed transfer protocol are supported. For efficient usage of PVP-SW feature and realization of high system performance, a dedicated compiler and system software are also under designing.

1 はじめに

CP-PACS (Computational Physics by Parallel Array Computer System) は，現在筑波大学にて開発が進行中の超並列計算機である．この研究は「学術の新しい展望のためのプログラム」(略称「新プログラム」，科学研究費補助金；創成的基礎研究費) 制度に基づき，"専用並列計算機による「場の物理」の研究" として平成4年度から5年間の計画で採択されたものである．その研究目的は**超並列計算機の開発**と，それを用いた**計算物理学の研究** (理論物理・実験物理と並ぶ第3の物理学研究方法) を行うことである．参加メンバは筑波大学，東京大学，慶応大学，京都大学，高エネ研，物性研，天文台からなる．本研究がスタートするにあたっては，筑波大学で先行して実施されていたPAX (Processor Array Experiment) シリーズ，特に **QCDPAX** の経験・成果が大きく寄与したことは言うまでもない．

このように，CP-PACSは新規に超並列計算機を開発するとはいえ，開発品は各種科学技術の実計算に実用することをも大きな目標の1つとしていること，そのため計算機工学関係者のみならずユーザやアルゴリズム開発関係者が共同研究を行っていること，に大きな特徴がある．

この研究推進のため，全国共同利用の「**筑波大学計算物理学研究センタ**」が平成4年度から設置されている．現在このセンタには，QCDPAXが移設され，またCP-PACSのフロント計算機 (FCS) などの一部も設置されてフル稼働中であり，その他の設備の整備も行われている．

超並列計算機CP-PACSそのものの開発に関しては，平成7年度中の稼働を目指して，当初基本的な検討を行い概略の目標仕様イメージを定め，その後アーキテクチャ・ハードウェア・ソフトウェアの基本設計・詳細設計を行いつつある．その間開発設計の各段階で，計算物理学の実問題についての仮想ベンチマーク評価結果と実現可能性とにより改善を行ってきた．関連した個々の中間成果に関しては従来も断片的に発表する機会はあったが，ここではCP-PACSとして纏めてそのアーキテクチャの概要を報告することにする．

なお具体的な装置などの製作については，当初製作委託を受ける可能性のあるメーカ16社 (内 海外6社) に全体計画概要を示して提案を求め，最終的に入札により (株) 日立製作所を選定している．

2 CP-PACS全体のアーキテクチャ

本節では，現状でのCP-PACS全体の主要目標仕様をまとめて先ず掲げ，次にそのアーキテクチャにつき簡単な説明を行う．さらに主要な項目についての詳細は次節以下に述べる．

2.1 CP-PACSシステムの主要目標仕様

(1) 全体の理論的なピーク性能：(64 ビットデータ) 300 GFLOPS 以上

(2) 全体の主記憶容量：64 GB 以上

(3) 並列方式：分散記憶型 MIMD 方式

(4) ノードプロセッサ (PU)：1024 ノード以上，PA-RISC 1.1 仕様のチップに PVP-SW 機能付加

- 物理浮動小数点レジスタ数：128
- クロック周波数：150 MHz 以上
- キャッシュメモリ L1：16 KB(I), 16 KB(D), L2：512 KB 以上

(5) 相互結合用ネットワーク：3 次元 Hyper-Crossbar network, 8×17×8 (1024 ノードの場合)

- 転送ピーク性能 ：200 MB/sec 以上
- 転送方式：wormhole 転送，高速プロトコル
- その他機構：バリア同期，放送，ブロックストライド転送

(6) 補助記憶装置：550 GB 以上, RAID-5, IOU (I/O ユニット) 経由 分散接続

(7) 全体を小部分に分割した運用：可能とする

(8) ソフトウェア：

- 言語：アセンブラ, C, Fortran, HPF(予定)
- ライブラリ：プロセス間通信用，高速ファイル処理用など
- OS：マイクロカーネルをベース

(9) 外部接続：

- FCS (front computer system) との接続：ピーク 100 MB/s 以上高速接続
- その他：IOU 経由 FDDI, Ethernet, (SCSI)

(10) 稼働開始：1995 年度

(11) 並列計算機システムの信頼度：ハードウェア MTBF > 2000 Hr

2.2 CP-PACSのアーキテクチャ

(1) のCP-PACSの目標性能については，計算物理関係者からの要望は表1にその例を示すように厳しいものであった．一方，最近の半導体技術の進歩および計算機設計技術の進歩により，PUとしては研究期間中に300 MFLOPS以上の性能のものの実現が充分可能と予測されたので，これを中心に考えることとし，コストパフォーマンス的配慮からも単体のPU性能を無闇に強化する方策は採らないことにした．

並列機を構成するPUの総台数については，PU台数も出来る限り大きくしたいが，その反面，

表 1: 計算物理学 大規模計算の所要能力評価例

仮定した性能	計算例 (主たる解法)	格子数	所要計算時間	所要主記憶容量	所要補助記憶容量
400 GFLOPS (実効)	quenched QCD (モンテカルロ法)	64^4	68 日	18 GB	216 GB
20 % (通信/演算)	full QCD (連立一次方程式)	24^4	58 日	1 GB	3 GB
300 MB/sec (I/O)		32^4	244 日	3 GB	9 GB
500 GFLOPS (実効)	宇宙流体力学 (オイラ格子法)	1000^3	120 hr	80 GB	400 GB
100 MB/sec (I/O)	電磁流体力学 (Roe MHD 法)	800^3	250 hr	136 GB	683 GB

- 設置された時の装置の規模および信頼度が現実的か否か,
- 性能と必要とされるコストの問題

なども勘案する必要がある. CP-PACS では全体の PU を論理的に 3 次元に配置するイメージから, 先ず 1024 台構成のものを中心に各種の検討を行った. その結果, 物理的な実装の可能性とその限界も考慮して HXB ネットワークの採用と, 1024 台構成のシステムを目標仕様とすることとした. 実現可能性という意味からは, 2048 台以上の構成のものまで可能であるが, 予算上の要因から上記のような線となったものである.

(3) の並列方式については, 物理的な構成から考えて, 主記憶装置は各 PU に分散して実装する必要があるので, 必然的に分散記憶型となる. 物理的に分散・論理的に共有メモリ型の並列機の研究が最近行われているが, 未だ実用には充分ではない. そこで CP-PACS では send/receive 型のメッセージパッシング方式を採用することとしている.

並列処理の方法に関しては, 特に初期においては, SPMD (single program multiple data) 方式での処理が多いものと思われ,「場の物理」に対しては, SIMD 型のものでも良いのではないかという議論もあるが, SPMD は SIMD とは異なる点もあり, また多種多様な応用にも応えられ, しかも OLTP (on-line transaction processing), データベース処理などの並列計算機工学上の問題にも対処できるようにするため, MIMD 方式を採用することとした.

(4) の PU については, 当初汎用の高性能マイクロプロセッサをそのまま使用することを考えていたが, 実際の計算物理学の応用での性能を検討した結果, 汎用のマイクロプロセッサそのままでは, 大規模計算の場合に充分な性能を発揮できない (キャッシュが有効でない問題によっては実効性能はピークの 15 %以下にもなってしまう) ことが判明した. そこで CP-PACS としては, 3 節に述べるような PVP-SW 方式を新たに考案し, メーカの協力も得て汎用のマイクロプロセッサの拡張を行うこととした. この機能の付加により, 各 PU は実際の問題の計算時にピーク性能の 70 %以上の性能を発揮できることが期待されている.

(5) の相互結合ネットワーク に関しては, 隣接した PU の間のデータの転送のみならず, 多様な科学・技術計算に適合したものを狙い, 各種の結合網につき検討を行った. 最近では結合網の主要部分は専用 VLSI で構成されることになるので, その VLSI の実現可能性, および相互結合網全体の実装方法と性能との関係を検討し, 3 次元の HXB (Hyper-Crossbar) ネットワーク (図 1) を採用することとした. 1 本の物理的な転送リンクの転送性能は, パイプライン的転送を行うことにより, 200 MB/sec 以上の転送性能を目論んでいる. ネットワークの詳細については 4 節に述べる.

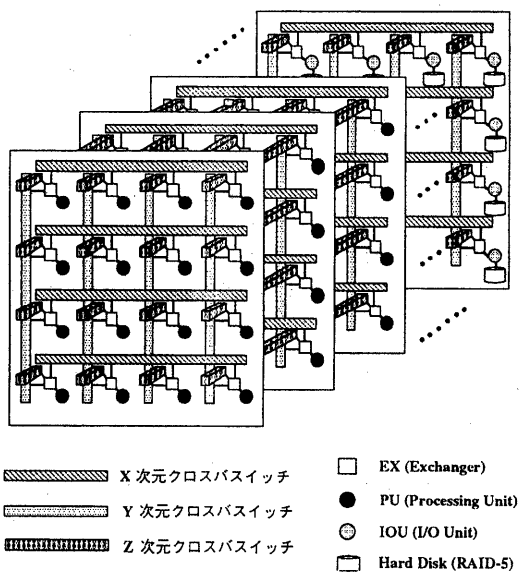

図 1: CP-PACS の全体構成

上記 (3), (4), (5) を基本とする並列処理方式の採用により, CP-PACS では計算物理学上の主要な計算 (例え

ばQCD計算)に関し，全体として実効速度(sustained speed)がピーク性能の50 %程度は達成できることを狙っている．

(6)の補助記憶装置については，長時間に及ぶ計算の中間結果などの一時記憶用，バックアップ用，その他のために，並列計算機から直接使用可能な総記憶容量500 GB以上の磁気ディスク記憶装置を分散して接続することとした．大容量のものであるので，信頼性を配慮して，RAID-5タイプのものを使用することとし．並列I/Oを有効に行うため，HXBネットワークにもう1面のIOU (input/output unit)プロセッサ平面を追加して，これ経由でRAIDディスクを分散接続することとした(図1参照)．

(8)のソフトウェア関係については，商用機のように完備したOS，言語処理系などのシステムソフトウェアを，CP-PACS向きに当初から揃えることは，開発力・予算から云っても無理であることは明らかであった．そこで必要最低限のシステムプログラムから開発することを方針とした．

最低必要となるプログラミング言語としてはFortran, C, およびアセンブラ言語のみを取り上げることとし，PUのアーキテクチャに追加されたPVP-SW機能をサポートしたコンパイラも実現することにしている．並列処理向きの言語としてはHPF (high performance Fortran)を取り上げることにしている．

OSに関しては，UNIX系のマイクロカーネルをベースとしたものを備える予定である．これを整備しつつ，結合ネットワーク通信用のライブラリ群を逐次整備していくことから運用が始まるものと考えている．

(9)のCP-PACSと外部との接続については，FCSとの間の高速大量のデータ転送手段としてピーク100 MB/s以上の接続を行い，ソフトウェアオーバヘッドを考慮した実効能力で50 MB/s以上の転送が可能なものとした．またジョブの投入・実行・出力の制御などためにFDDI, Ethernetなどの接続も行う予定である．

3 擬似ベクトルプロセッサPVP-SW

方針 汎用のRISCプロセッサの高速性は「キャッシュが有効に働く」という仮定の下に成立する．しかし，大規模な科学技術計算においては，データ領域が非常に大きく，またデータの時間的局所性が少ないという性質があるため，データキャッシュのミス率が大きく，プロセッサの性能は主記憶アクセスレーテンシのペナルティのために大きく低下する．

一方，科学技術計算において高い処理性能を有するベクトルプロセッサは，以下の特徴を持つ．(1) 主記憶アクセスがパイプライン化さ れ，主記憶のスループットが高い．(2) 多数のベクトルレジスタを持つためベクトルロード／ストア処理のベクトル長を長くできる．(3) チェイニング機構によりロード／ストア処理と演算処理とを並行に行える．これらの特徴により，ベクトルレジスタへのプリロード機能を実現でき，主記憶アクセスレーテンシが性能に与える影響を隠蔽することができる．

そこで，CP-PACSではこれと同等な機能／特徴をスーパスカラ方式のプロセッサに機能拡張を施して実現する．この方式を我々は**擬似ベクトル処理** (Pseudo Vector Processing)[1][2]と呼ぶ．そのためには，以下の3点を実現する必要がある．

- 多数の浮動小数点レジスタ
- レジスタへのプリロード機能の強化
- 主記憶アクセスのパイプライン化

我々の方針は，ソフトウェア全体の開発コストを抑えるべく，既存のRISCアーキテクチャとの上位互換性を保つことである．しかし，第1の点を実現するためには，命令フォーマット中のレジスタ指定フィールドのビット数を増やす必要があり，通常は命令セットアーキテクチャの互換性は失われる．そこで，我々は「擬似ベクトルプロセッサPVP-SW (Pseudo Vector Processor based on Slide-Windowed registers)」を提案した[3][4].

PVP-SWのアーキテクチャ PVP-SWのアーキテクチャは既存のスカラプロセッサに対する拡張として定義される．拡張点は，浮動小数点レジスタのスライドウィンドウ化と命令追加である．

(1) スライドウィンドウ構成

図2に，浮動小数点レジスタのスライドウィンドウ構成を示す．物理的なレジスタ空間は複数の論理的なウィンドウに分割される．1つの論理的なウィンドウ内のレジスタ数は，拡張前のアーキテクチャで定義される数(図2では32とした)と等しくなければならない．

ある時点ではただ1つのウィンドウのみがアクティブであり，このアクティブウィンドウを指定するポインタ**FWSTP**(**F**loating-point **W**indow **ST**art **P**ointer)を導入する．拡張前のアーキテクチャにおける命令は全てアクティブウィンドウ内のレジスタのみを用いる．一方，ソフトウェアでFWSTPの値を変更することにより，このアクティブウィンドウを物理的ウィンドウ空間内で自由に移動(スライド)することが可能である．ウィンドウ内のレジスタ数は拡張前のアーキテクチャに依存するが，物理レジスタの数はハードウェアが許す限り任意の個数実装することが可能であり，CP-PACSでは128個を予定している．このようにして，上位互換性を維持しながら多数のレジスタが利用可能となる．

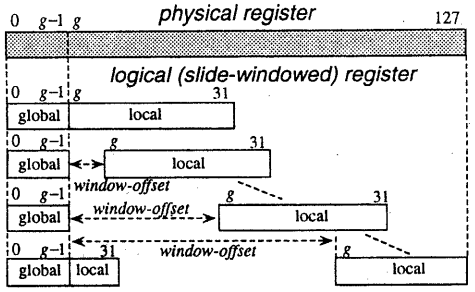

図 2: PVP-SW における浮動小数点レジスタの構成

(2) 追加命令
以下の命令が追加される.

- **FWSTPset** : FWSTP の値を変更し，アクティブウィンドウを切り替える.
- **FRPreload** : データを，任意に指定されるウィンドウ内のレジスタにメモリからロードする.
- **FRPoststore** : データを，任意に指定されるウィンドウ内のレジスタからメモリにストアする.

FRPreload 及び FRPoststore 命令は主記憶とレジスタとのデータ転送を行うが，以下の 3 つの特徴を持つ.

1. 「置いてきぼり処理」により，データ転送の完了を待たずに後続命令は実行される
2. アクティブウィンドウに関係なく，全物理レジスタをデータ転送の対象として指定できる
3. キャッシュミス時には主記憶とレジスタの間で直接データを転送し，キャッシュ内データを変更しない

1 番目の特徴により，将来必要となるデータに対するプリロード命令を先行発行することで，他の命令実行を妨げることなく該当データに対する主記憶アクセスレーテンシを隠蔽できる．より長いレーテンシを隠蔽するためにはより先行してプリロード命令を発行せねばならず，必要な物理レジスタ数が増加する．この問題は物理的に多数のレジスタを用意すること，及び 2 番目の特徴で解決する．3 番目の特徴は，キャッシュのスループットが十分で無い場合に性能が低下するのを防ぐため，及びキャッシュコヒーレンシ問題に対処するためである．これらの特徴により，PVP-SW 方式が主記憶アクセスレーテンシを効果的に隠蔽し極めて高い性能を示すことを，各種ベンチマークの評価によって確認している [5].

CP-PACS で採用予定のプロセッサ　CP-PACS では Hewlett Packerd 社の PA-RISC 1.1 アーキテクチャのチップにこの PVP-SW 機構を追加する．PA-RISC 1.1 は 32 本の倍精度浮動小数点レジスタを持っており，これを拡張して物理レジスタ群とウィンドウ制御回路 (主に論理レジスタ番号と物理レジスタ番号の変換をする) を追加することにしている.

4 ハイパクロスバ・ネットワーク

HXB とその特徴　CP-PACS における PU 及び IOU 間の結合は Hyper-Crossbar Network(以下 HXB) を用いる．HXB のトポロジは n 次元の超立方体の概念を拡張した形としてとらえることができる．ここで次元数 n は任意に選択可能であるが，CP-PACS 実装時の技術を想定すると，3 次元の構成が妥当であると考えられる.

一般的な 3 次元 HXB の特徴について，$X \times Y \times Z$ 構成の場合を想定して説明する (図 1は 4×4×n の構成)．まず，全 PU は $X \times Y \times Z$ の 3 次元正方格子状に配置される．そして，1 つの次元方向 (例えば x 次元方向) に並んだ PU 同士をクロスバ・スイッチ (以下，XB と省略する) で完全結合する．例えば x 次元方向の場合，X 入力 X 出力 の XB が必要となる．このようにして全ての次元について，その次元方向に並んだ PU 間を XB で結合する．この場合，次元数は 3 なので，各 PU は 3 つの次元方向の XB に対するリンクをそれぞれ持つことになる.

メッセージ転送における固定方式ルーティング・アルゴリズムは単純である．$PU(x_s, y_s, z_s)$ から $PU(x_d, y_d, z_d)$ までメッセージを転送する場合，まず $PU(x_s, y_s, z_s)$ から $PU(x_d, y_s, z_s)$ までメッセージを送り，続いてそれを $PU(x_d, y_d, z_s)$ へ，そして最後に $PU(x_d, y_d, z_d)$ へ，というように 3 ステップの転送を繰り返して送ればよい．CP-PACS ではこの際に転送遅延を極力小さくするために Wormhole 方式のメッセージ転送を行う．このため，一つの PU と各次元方向の XB を結合するために，EX (Exchanger) と呼ばれるルータ・スイッチを設ける．EX 自身も小規模の XB である (例えば 3 次元の場合，EX は 4×4 の XB である).

Wormhole 方式の HXB には,

- ネットワークの半径が小さい
- 同一サイズあるいはそれ以下のサイズのトーラス・ネットワークをエミュレートできる
- Binary-n-Cube ネットワークのエミュレートが容易である
- ブロードキャスト転送が容易に実現できる
- ランダム転送におけるスループットが非常に高い

等の特徴がある [6].

Wormhole 方式の HXB では，一般的に経路決定を任意に行うとデッドロックが生じが，各通信チャネルに次

元数nと等しい数のバーチャル・チャネルを用意すれば，経路決定を動的に行う適応ルーティングが可能であることも知られている[7]．しかし，CP-PACSにおいては，科学技術計算における定型的なPU間交信がアプリケーションの多くを占めることと，通信チャネルをバーチャル化するためのコストを考慮して，固定ルーティング方式を用いる．

以下に，CP-PACSにおけるPU間結合ネットワークの特徴をいくつか挙げる．

高速転送モード　CP-PACSでは高速なメッセージ転送立ち上げを実現するために，ユーザ・アプリケーションから直接ネットワークへのメッセージ転送起動が行えるような機構を提供する．この機構を用いる転送を高速転送モードと呼ぶ．高速転送モードでは，NIA (Network Interface Adapter) へのメッセージ送出命令をユーザ・モードから直接発信することが可能となっており，これによって，システム・コールによるオーバヘッドなしに，メッセージの送出が可能となっている．送出されたデータは受信PUのユーザ空間に直接書き込まれるが，SPMD方式のプログラムであれば送信側でもそのアドレスがわかるので問題は生じない．高速転送モードは，アプリケーション・レベルで実行可能な一種のPU間DMA転送と考えることができる．

ブロードキャスト転送　並列処理においてしばしば現れる特殊なメッセージ交換の例としてブロードキャスト通信がある．CP-PACSではこれをハードウェア的にサポートする．HXBにおいては，あるXBの入力に届いたメッセージを，そのXBの全出力に同時に送出することにより，単一次元内でのブロードキャストが容易に実現できる．これを全次元に対して行えば，全PUに対するブロードキャストが実現できる．ただし，HXBでは複数PUからの別々のブロードキャスト要求はデッドロックを生じるため，例えばあるPUだけがブロードキャスト・メッセージを流すことができるという制限を設ける必要がある．現在は基本的にこの手法を元に設計を行っている．

ブロック・ストライド転送　一般に，多次元配列データの一部をPU間で交換するような場合，単純な連続領域のデータ転送ではうまく行かない場合が多い．このような場合にはブロック・ストライド転送が用いられる．高速転送モードを用いた場合，ストライド転送をOSのメッセージ・バッファを用いてソフトウェア的に処理することは不可能なため，これはNIAで自動処理されるようになっている．NIAではストライド・アクセスをしたデータをまとめてメッセージとして転送するため，ネットワーク転送に要する時間は実際に送信するデータ量の分だけで済む．

バリア同期機構　一般的に，メッセージ・パッシング系のPU間交信では，メッセージの送受信処理そのものがpoint-to-pointの同期操作を同時に実現しているため，その他の同期処理が不要な場合が多い．しかし，現実的には並列プログラムのデバッグ時や，システムの初期化等において，一斉同期が必要な場合も多い．そこで，CP-PACSではこのような場合に簡単に全PUでバリア同期をとることができるような機構を用意している．CP-PACSにおける同期ネットワークは，基本的にデータ転送ネットワークを部分的に用いることにより実装される．これは各XBにおいて同期用メッセージの待ち合わせと合流を制御することによって実現される．

5　ソフトウエア・システム

方針　ソフトウェア・システムとして持つべき基本的なものは，OSとプログラミング環境である．CP-PACSのような超並列計算機システムでは，実際のアプリケーションプログラムで高性能を実現出来ること，そのようなプログラムが比較的容易に作成可能であること，の2点が重要である．そのためには，OSは，(1) OSの関与が性能に悪影響を及ぼさないように軽いOSであること，(2) 柔軟な並列実行形態が指定出来ること，(3) PU間のデータ転送が高速に行えること，(4) 複数のIOUでの並列I/Oによる高速なファイル処理が行えることが重要である．プログラミング環境は，(5) 一般によく使われている環境に近いものであること，(6) 高級言語での並列プログラムの作成やデバッグのしやすい環境であること，(7) 高性能を実現するために機械語レベルでのプログラミング環境もあること，などが必要である．

並列実行OS機能　各PUの基本OSはUNIX系のMachをベースにし，並列実行のために以下の機能を設ける．

1. PU分割の機能：たとえば，大規模並列計算用の大PU集合と，デバッグ用の小PU集合に分ける．分割されたPU集合はパーティションと呼ぶ．この分割の指定はシステム管理者が行う．
2. プロセス生成の機能：ユーザプログラムは一つのパーティションの中で実行される．ユーザプログラムでは，そのパーティションの中のPU群を指定し，その各PUの中にプロセスを生成することが出来る．

ファイルに関しては，通常のUnix 4.3 BSDと互換性のあるファイルシステムであること，並列機全体で一つのツリーをなすファイルシステムであること，を

基本とする．さらに，PU が 3 次元の配列になっており，IOU 平面を持つというシステム構成を最大限に活用して，ファイル入出力の並列実行を実現するために，(1) 一つのファイルを複数の IOU に分散配置する機能，(2) 各 PU から HXB 上でそれに最も近い IOU に分散配置されている (あるいは分散配置しようとする) ファイルの位置を求める機能，を設ける．

プロセス間通信機能　通常の，OS を介した通信機能の他に，OS をほとんど介さない特別に高速な通信機能を設ける (4 節参照)．

プログラム言語　プログラム言語としては，アセンブラ言語，C，Fortran，HPF (High Performance Fortran) を備えることとした．超並列計算機用のプログラムを書くのは容易ではない．メッセージ通信などを意識せずにプログラムを書く一般ユーザには，通常の Fortran プログラムにデータの分散配置を指定するだけで並列プログラムが出来る HPF のような言語が必要である．エキスパートには，C や Fortran で並列を意識したプログラムを書き，それに並列に実行されるプロセス間でのメッセージ通信の命令 (手続き呼び出し) を入れられるようにする必要がある．いずれの場合も，並列計算機の性能を最大限に引きだすにはコンパイラによる徹底した最適化が必要であるが，コンパイラによる最適化にも限度があるのでアセンブラ言語でのプログラミングも可能とし，そのプログラムのデバッグシステムも考慮する必要がある．以上がこれらの言語を備えることにした理由である．なお，これらの言語についてはワークステーション上のクロスコンパイラと実機上のセルフコンパイラの両方を開発する予定である．

並列計算機の性能を最大限に発揮させるためにはコンパイラによる最適化がその鍵になる．並列プログラムの最適化として PU 間通信の最適化が重要であることは言うまでもないが、高い数値演算性能を実現するためには個々の PU 上で動くプログラム (オブジェクト・コード) 自体が高度に最適化されていることも重要である (6 節参照)．

プログラム開発・デバッグ機能　プログラム開発用のツールとしては，エディタ，コンパイラ，デバッガ，性能モニタなどがある．

リモートデバッガは，ワークステーションなどをホストとして，並列計算機 (ターゲットマシン) のプログラムをデバッグするためのものであり，ホストマシン上から並列計算機上のプログラムの実行制御をしたり，メモリやレジスタの内容の参照や更新が出来る．

トレーサ・モニタは，並列計算機上のプログラムの実行の履歴 (トレース) をとるものであり，その結果をワークステーションなどで解析して性能評価等を行うためのものである．実行の履歴としては，各機械語命令のトレース，システム・コールなどのイベントのトレース，PU 間通信のトレース，などがある．

シミュレータは，並列計算機用のプログラムをソースプログラムの変更なしに，ワークステーション上で走らせるシステムである．そのためにスライドウィンドウの機能と CP-PACS 特有の通信機能をシミュレートする機能を持たせる．

6　PVP-SW 用のコード最適化

ここでは各 PU 上で実行されるオブジェクト・コードの最適化，特にループ最適化について述べる．2 節の擬似ベクトルプロセッサは既存の RISC プロセッサの上位互換であり，我々のループ最適化技法はいわゆるソフトウェア・パイプライニング (software pipielining)[8]，あるいはそれをアルゴリズム化したモジュロ・スケジューリング (modulo scheduling) を基本としている．ソフトウェア・パイプライニングとは，ループ実行の命令列を複数のステージ (あるいはフェーズとも呼ぶ) に分割し，それらステージをパイプライン的に同時実行する，「置いてきぼり処理」を巧みに利用する方法である．これによって命令のレーテンシを隠蔽し，プロセッサ・リソースを効率的に利用できる．たとえば以下のループ:

```
DO I = MIN,MAX
   X[I] = A * Y[I] + B
ENDDO
```

は次のようなオブジェクト・コードへ変換される．実際のコードには前処理 (prologue) 部，本体 (kernel) 部，後処理 (epilogue) 部があるが，ここでは本体部のみを示す．ただし，命令並列度は 2，FRPreload，乗算および加算命令のレーテンシはそれぞれ 10，2，2 とし，定数 A,B はスライドしないレジスタ R(4),R(5) に事前に格納されているとする．

```
1 : FRPreload Y[I + 6], R(35)    R(31) := R(4) * R(31)
2 : FRPoststore R(29), X[I]      R(30) := R(5) + R(30)
3 : FWSTPset 1                   ConditionalBranch
```

上のコードでレジスタ番号の一連の変化

$$R(35) \rightarrow \ldots \rightarrow R(31) \rightarrow R(30) \rightarrow R(29)$$

がレジスタのスライドを表している．

任意に与えられたループ・プログラムからこのようなオブジェクト・コードを得るには，命令のスケジューリングとレジスタ割付の二つの処理を行わねばならないが，この解法は次の二つに大別される．ひとつは，ま

ず命令スケジューリングを行い，その後にレジスタ割付を行う方法 (pre-pass scheduling) である．もうひとつは逆に，まずレジスタ割付を行い，次に命令をスケジューリングする方法 (post-pass scheduling) である．スライドウィンドウを利用する場合には論理レジスタ番号が動的に変化するため，後者の方法を適用することは難しいと考えられ，著者らは前者の方法に基づいてコード最適化器を試作した．この前者についてさらに詳細に述べると，以下の手続きとなる．

1. データ依存グラフの作成
2. 命令の優先順位の決定
3. ループ立ち上げ間隔の決定
4. 空なリソース予約テーブルの作成
5. 優先順位に基づくリソース予約テーブルへの命令の埋め込み
6. レジスタ干渉グラフの作成
7. レジスタ干渉グラフの彩色

上の 1., 2., 4. の手続きはスライドウィンドウを用いることに特に関係せず，従来と同様である．3., 5. は以下の点で従来よりも容易になる．6., 7. は従来法からの拡張が必要である．

まず，繰り返し処理の度に論理レジスタ番号が一定の刻み幅で変化していくから，複数のパイプライン・ステージで使用するレジスタ番号の間の干渉が容易に回避できるという利点がある．いわゆる register renaming がそのまま実現されているのである．通常のソフトウェア・パイプライニングでは，レジスタ間の干渉のため，ループ立ち上げ間隔はそのループ処理に含まれる命令の中の最大のレーテンシ (多くの場合，主記憶アクセスレーテンシ) 以下にはできない．この対策として一般には loop unrolling を行い，かつ命令スケジューリングをより巧妙にする．スライドウィンドウを用いる場合にはそのような技巧を凝らす必要がないため，上記 3. ループ立ち上げ間隔の算出や，5. 命令スケジューリングのアルゴリズムがごく自然な発想に基づいて容易に実現できる．

論理レジスタ番号が変化していくため，レジスタ干渉グラフも従来のものとは異なる．従来のレジスタ干渉グラフではひとつの頂点はひとつの物理レジスタを表すが，我々の場合にはひとつの物理レジスタには複数の論理番号が対応する．そのためレジスタ干渉グラフでは複数の頂点でひとつの物理レジスタを表す等の拡張が必要となる．よってレジスタ彩色アルゴリズムも従来よりも複雑になる．彩色問題が NP 問題である点では従来と同様であるが，有効な近似アルゴリズムについては今後の課題である．

現在，著者らはループ本体が基本ブロックだけからなる単純な場合についてコード最適化器を試作し，種々のアルゴリズム実験を行っている．今後，条件分岐を含むループ [9] や多重ループについても検討していく予定である．

7 おわりに

CP-PACS は，計算物理学の問題を処理するために筑波大学で開発中の超並列計算機である．勿論一般の科学技術計算用をも目指したものである．本報告では，CP-PACS が目指すアーキテクチャの概要と，特徴的な擬似ベクトルプロセッサ方式，ハイパクロスバ結合網，計画しているソフトウェアの概要について報告した．

CP-PACS の開発は平成 7 年度中の稼働開始を目指し，現在詳細な設計が進行中である．一方実計算での性能評価仮想ベンチマーク，各種のシミュレータによる評価，並列計算用アルゴリズムの検討，応用プログラムの検討も行われている．

本研究は文部省科学研究費・創成的基礎研究費 (06NP0601) によって行われているものである．本研究を推進するに当っては，文部省・関係大学・関係機関・筑波大学の広い範囲の関係者のサポートをいただいている．他の同様プロジェクトの方々からも有益な御示唆をいただいている．ここに感謝の意を表します．また日頃協力をいただいている (株) 日立製作所の関係者にも謝意を表します．

参考文献

[1] K.Nakazawa et.al., "Pseudo Vector Processor based on Register-Windowed Superscalar Pipeline", Proc. of Supercomputing '92, pp.642-651, Nov., 1992

[2] 中村宏 他, "レジスタウィンドウ方式を用いた擬似ベクトルプロセッサの評価", 情報処理学会論文誌第 34 巻 4 号, pp669-680, 1993 年 4 月.

[3] 位守弘充 他, "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第 34 巻 12 号, pp2612-2623, 1993 年 12 月.

[4] H.Nakamura et.al., "A Scalar Architecture for Pseudo Vector Processing based on Slide-Windowed Registers", Proc. of ACM/IEEE ICS '93, pp.298-307, Jul., 1993

[5] H.Nakamura et.al., "Evaluation of Pseudo Vector Processor based on Slide-Windowed Registers", Proc. of HICSS-27, pp.368-377, Jan., 1994

[6] 朴泰祐 他, "ハイパクロスバ・ネットワークの性能評価", 信学技報 CPSY93-40, 1993 年 11 月.

[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 予稿集, pp.129-136, 1994 年 5 月.

[8] Lam. M. S., "A Systolic Array Optimizing Compiler", Kluwer Academic Pub., 1989.

[9] 山下義行 他, "ループ中に条件分岐を含む場合の最適なソフトウェア・パイプライニング", 並列処理シンポジウム JSPP'94 予稿集, pp.17-24, 1994 年 5 月.