特　集「第五世代コンピュータ」

並列推論マシン PIM

The Parallel Inference Machine (PIM)

市吉　伸行*　後藤　厚宏*　近山　隆*
Nobuyuki Ichiyoshi　Atsuhiro Goto　Takashi Chikayama

＊（財）新世代コンピュータ技術開発機構
Institute for New Generation Computer Technology

1989年2月10日　受理

Keywords: parallel inference machine, concurrent logic language, metaprogramming, network-connected multiprocessor, parallel implementation, parallel operating system.

1. はじめに

第五世代プロジェクトは，1990年代の大規模知識処理システムのベースとなる第五世代コンピュータの開発を目指しているが，その特徴は，並列マシンと知識処理システムを結ぶ層として論理型の核言語を設定し，そこから下層および上層を設計していくインサイド・アウトのアプローチを取っているところにある．このアプローチをとる大前提として，①大規模知識処理システムの要求する計算能力にこたえるためには並列実行が必須であること，②複雑な知識処理システムを見通し良く設計するためには明確な意味論を持つ論理プログラミングが有効であること，③論理プログラムは並列処理に向いていること，という認識がある．

本稿では，第五世代コンピュータシステムのエンジンとなる並列推論マシン（ハードウェア＋言語処理系）とその上の並列オペレーティングシステム PIMOS について解説する．初めに並列論理型言語 KL1 の設計思想および言語仕様を述べ，次に PIM のハードウェアとその上の KL1 処理系の実現方式を概説し，PIMOS については，その機能・特徴と OS 保護の手法を紹介する．最後に，研究開発の現状と課題について触れる．

2. 並列論理型言語 KL1

並列論理型言語 KL1 は，ストリーム AND 並列型言語 GHC [10] に制限を加えた Flat GHC (FGHC) をベースにして，OS を支援し，効率的並列実行を可能にするための実行管理，資源管理，優先順位および負荷分散指定の諸機能を追加したものである．

2・1 Flat GHC

Flat GHC (FGHC) は，GHC に対してガード部に限られた組込み述語のみの出現を許すような制限を加えたものである．FGHC ではガード実行がネストせず，AND/OR 木や変数レベルの管理が不要なので，オーバヘッドの小さい処理系が実現できる．ガードにユーザ述語が書けないことにより表面的記述力は落ちるが，ガードにおけるテストをボディ・ゴールの組合せに変換できるので，本質的記述力は低下しない．

2・2 荘　　園

荘園とは，実行制御と資源管理を行うためのメタプログラミング機能である．末尾再帰呼出しされる KL1 のゴールが小粒度プロセスを表すのに対し，荘園は一つのまとまった計算を定義し，荘園単位に実行の監視や制御が行えるようになっている．荘園は組込み述語 execute の呼出しによって生成される．

execute (*Goal, Control, Report, Mask*)

ここで，*Goal* は KL1 データとして表現された KL1 ゴール，*Control* は実行制御用のコントロールストリーム，*Report* は実行監視用のレポートストリーム，*Mask* はこの荘園で捕捉する例外事象を指定するマスクである．生成された荘園の下で *Goal*（およびその子孫ゴール）が実行される．

荘園の下で新たに execute を呼び出すことによりネストした荘園（子荘園という）を生成することもできる．

コントロールストリームに *start, stop, abort* などのコマンドを流すことにより，実行の開始（再開），中断，放棄などができる*.

レポートストリームには，荘園内で起きた特定の事象を報告するメッセージが流れる．荘園の終了（すべてのゴールおよび子荘園が終了したこと）の報告，例外事象の発生の報告などである．

荘園は資源管理の単位でもある．資源とは計算量の目安であり，計算時間やメモリ消費量と大体の比例関係にある．荘園があらかじめ指定された量の資源を消費してしまうと実行中断状態になり，資源の不足がレポートストリームに報告される．

荘園の実行管理プログラムは，コントロールストリーム・コマンドにより，消費可能資源の上限を増やすことができる．資源管理機構により，プログラムの暴走を防いだり，荘園単位のフェアなスケジューリングが可能になっている．

KL1 の例外事象には例外の種類によって決まるタグ値が付随しており，例外事象の起きた荘園およびその祖先荘園のうちで，タグ値とマッチするマスク値を持つような一番内側の荘園のレポートストリームに次の形で報告される．

exception (*Type, Goal, NewGoal*)

ここで，*Type* は例外の種類を表すデータ，*Goal* は例外が発生したゴール，*NewGoal* は未束縛変数である．荘園の実行管理プログラムは，上記メッセージを受け取ると *Type* および *Goal* を解析し，*NewGoal* を適切なゴールで具体化することによって実行を継続したり，場合によっては *abort* コマンドで実行を放棄することもできる．タグ値の機構により，各荘園は選択的に例外処理ハンドラを書くことができる．

例外事象には，ゴールの失敗（すべての節のガードが失敗した），ボディ・ユニフィケーションの失敗，組込み述語例外（不正な入力引数，算術演算例外など），raise 組込み述語呼出しによるソフト例外などがある．これらはソフト例外を除けばすべて FGHC プログラムにおける失敗である．

例外処理機能は，OS サービス提供や KL1 の言語機能拡張に使うことができる．例えば，整数の 3 と構造型データ 1＋2 のユニフィケーションの失敗時に再開ゴールとして true を与えることで（オーバヘッドはかなり大きいが），ユニフィケーションのセマンティックスを拡張できたりする．

* KL1 におけるストリームは，メッセージを要素とするリストとして表される．メッセージの送り手と受け手の間の同期は KL1 の同期機構によって自然に実現される．

2・3 優先度指定・負荷分散指定

優先順位および負荷分散は，ボディ・ゴールにプラグマと呼ばれる次のような標記を付けることによって指定する．

Goal@priority (*Prio*)

Goal@processor (*Proc*)

前者は *Goal* を優先度 *Prio* で実行せよという指定であり，後者は *Goal* をプロセッサ *Proc* で実行せよという指定である．プラグマはプログラムの論理的意味を変えない．

荘園ごとに実行優先度の上下限が決まっており，実行優先度は，荘園の優先度幅の中の割合で指定したり（絶対指定），親ゴールの優先度と上下限の間の割合で指定したり（相対指定）できる．優先度機構の主な目的は，ヒューリスティックな探索順序のようにデータ依存型でない実行順序を指定するためである．そのために物理的に 2^{12} 程度の細かい優先度レベルを提供するが，応用プログラムの実行効率を上げるのにこの細かい優先度指定が役立つことが示されている[12]．また，ユーザプログラムの暴走のために OS プロセスがなかなかスケジュールされないといった状況は，OS プロセスの優先度をユーザプログラムよりも高くすることで防げる．

MIMD 型並列マシンの性能を十分に引き出すためには，実行負荷をよく分散しなければならない．しかし，単純な負荷の均等化はプロセッサ間通信オーバヘッドを大きくするし，KL1 プログラムが対象とする非定型な問題では計算負荷の事前の予想が難しい，という困難がある．種々の負荷分散手法を試すために，現在のところ上記のように低レベルの指定法を提供している．

3．並列推論マシンのハードウェア

KL1 を実行する並列マシンが Parallel Inference Machine (PIM)[3] である．現在，ハードウェア詳細仕様の少しずつ異なるいくつかの PIM を設計開発中だが，それらは KL1−B[6] と呼ばれる共通の中間言語をサポートする．ここでは，そのうちの一つである PIM/p[14] を紹介する．

PIM/p のアーキテクチャは，図 1 に示すように階層的である．すなわち，下位のレベルとして 8 台の要素プロセッサがメモリを共有するクラスタがあり，上位のレベルとして複数のクラスタが高速ネットワーク（ハイパーキューブ）で結合され，全体として当初

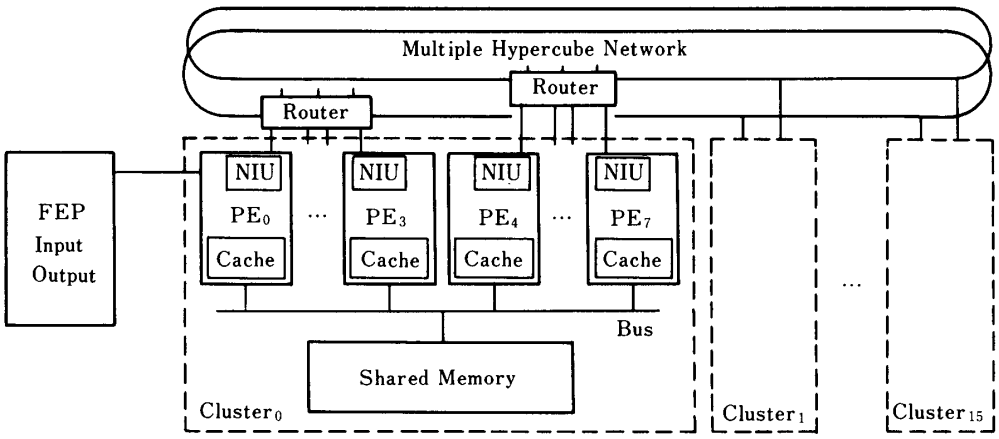


図 1　PIM/pのアーキテクチャ

128プロセッサの並列マシンが構成されるが，将来的には1000台程度の規模まで拡張可能となっている．

メモリ共有型アーキテクチャは共有バスがボトルネックになるため，要素プロセッサ10数台程度が限界と見られ，大規模並列マシンを実現できない．また，大規模ネットワークにおいては，配線が可能であるためには，何らかの局所性を持つ（つまり，プロセッサ間の距離に遠近がある）ような結合方式とならざるを得ない．これが，上位構造が局所性を持つネットワーク型になっている理由である．

一方，プロセッサ間の通信はメッセージ通信よりも共有メモリを介した通信のほうがコストが安い．PIMの典型的アプリケーションは，多くのプロセスが部分を受け持って全体として一つの問題を解くような通信の多い型のものが予想されるので，台数性能比を上げるために通信コストはできるだけ抑えたい．下位レベルに密結合のクラスタを採用した理由は，相互間通信の多いプロセス群を同一のクラスタに割り当てることによって，通信オーバヘッドを低減できることにある．

プロセッサは並列記号処理向きに，タグ処理機能や後述のMRBの支援機能を持ったRISC型パイプラインアーキテクチャのものを新規設計した．KL1−B命令のうちの大半の単純なものは短いサイクル時間で実行されるが，複雑なケースの処理はマクロ命令メカニズムにより，プロセッサ内蔵命令メモリ中のルーチンに割り出されて実行される．これよりオブジェクトコードの静的サイズが抑えられ，シミュレーションの結果，キャッシュヒット率の上がることが期待されている[9]．

クラスタ内では，分散したキャッシュ間の一貫性を保つことや競合資源のロックを低いオーバヘッドで実現することが必要である．そのために，KL1プログラムの実行特性に合わせたバス・プロトコルを持つ一貫性キャッシュを設計した．例えば，プロデューサの書いたデータを別のプロセッサ上のコンシューマが読む場合にキャッシュ間データ移動が起こるが，ほとんどの場合，プロデューサはそのデータに再びアクセスしないので，移動元のキャッシュエントリのデータを無効化（開放）できる．また，共有されていないことのわかるデータは，バス・サイクルを使わずにロックできる．

ネットワーク接続装置（NIU）には，バイト/ワード変換とパケット・バッファリング機能，およびフロント・エンド・プロセッサ（FEP）やディスク装置などの入出力機器を直結するための汎用入出力インタフェースが用意されている．PIM上のアプリケーションの多くは計算バウンドだと思われるが，自然言語処理などでPIMの大きな実メモリにも入り切らないような大きなデータベースを必要とするような場合は，多数の入出力機器を各プロセッサに分散して接続することにより，システム全体の入出力バンド幅を拡張することが可能である．クラスタ間のネットワークは，各クラスタをノードとするハイパーキューブ網を二重化したものである．全プロセッサはNIUを介してどちらかの網に接続されている．

4．KL1処理系

PIMの上でKL1プログラムの実行を司るのがKL1処理系である．KL1処理系の開発において，論理型言語分散処理系に特有の問題，すなわちプロセッサをまたがって実行される荘園における実行制御や終了検出の実現，分散ユニフィケーションの実現，クラスタをまたがる参照ループ生成の防止などの問題の解決に新しい技術の開発が必要であった．

またKL1は，破壊的書込みの許される言語やバックトラックによりメモリが自動的に解放されるPrologと比べてメモリ消費速度が速いので，トータルな実行

性能を高く保つためには効率の良いガーベジ・コレクタ (GC) が必須であり，さらにクラスタをまたがるGCも必要であった．

以下，KL1の基本実行モデルを配述した後，個々の問題点への解決策を紹介する．

4・1 基本実行モデル

我々の処理系は，以下のような実行モデルに従っている．AND関係で結ばれたゴールの集まりであるゴール・プールとプロセッサの集合があり，各プロセッサはゴール・プールからゴールを取り出して，リダクションを試みる．プロセッサの数が増えると，同時に多数のゴールのリダクションが行えるので，処理系のスループットが上がることになる．

これが基本実行メカニズムであるが，実際には，KL1ではゴール・プールは荘園に対応し，ゴールの失敗は例外の形でゴール・プールの外に伝えられる．

KL1プログラムは，プログラム・モジュール単位にWAM*1レベルの中間言語KL1−Bにコンパイルされる．コンパイラはレジスタ・レベルの最適化やコミットできる節を高速に見つけるクローズ・インデキシング・コードの生成を行っている．

4・2 荘園里親方式

クラスタをまたがった荘園においては，分散実行制御や分散終了検出が大きな問題となる．荘園には，実行可能状態や中断状態があり，もしその情報を集中化しているとゴールがスケジュールされるたびにそのゴールの荘園を管理しているクラスタに状態を問い合わせねばならず，オーバヘッドが大きい．そこで我々の処理系では荘園の管理情報をクラスタ間に分散し，それを里親と呼んでいる[4]．

クラスタ内では，共有データ参照（特に書込み）は明示的なロック/アンロックが必要なだけでなく，キャッシュ間データ転送のためのバス使用を伴い，処理性能がバス性能で抑えられる要因となる．これを緩和するために，フリーリストや実行可能ゴールリストをプロセッサごとに個別に持つことによって，データやゴールへのアクセスをできるだけ局所化している[8]．

分散した荘園における実行終了の検出は，いわゆる分散計算終了検出の問題であるが，重みつきレファレンス・カウントを応用したWTC方式[7]を採用した．簡単に言うと，分散された個々の里親とゴール投げ出し用メッセージに正の重みを分け与え，各里親における実行の終了に伴って重みを回収し，すべての重みの回収をもって全体の実行終了を検出しようというものである．これにより，以前の方法と比べて，分散実行のためのメッセージ数が4分の1程度削減できた．

4・3 分散データ管理と分散ユニフィケーション

分散処理系で全体を一つのアドレス空間とすると，クラスタごとの独立したGCが難しくなるという難点がある．クラスタ内GCによりデータのアドレスが変わると*2，そのデータを参照しているすべてのポインタを更新しなければならないが，独立して動いている他のクラスタ内のポインタの更新は非常に難しい．

そこでKL1処理系では，クラスタ内外でアドレス系を分け，各クラスタはクラスタ外からクラスタ内への参照を管理する輸出表と，クラスタ内からクラスタ外への参照を管理する輸入表という二つのアドレス変換テーブルを管理することとした．

データがクラスタ間に分散していると，ユニフィケーションにおいてクラスタ外のデータを読んだり，書き込んだりする場合が出てくるが，そのためのreadプロトコルとunifyプロトコルを設計した．また，分散ユニフィケーションを無秩序に行うと外部参照ポインタからなるループができる可能性があるので，クラスタ番号を比較して番号の大きいクラスタを指す外部参照は自クラスタの変数にバインドしてはいけない（その代わりにunifyメッセージを出す）という規則を設けた*3．

4・4 ガーベジ・コレクション

KL1処理系におけるGCの重要性から，クラスタ内・クラスタ間の即時および一括型のGCが開発された．

MRB方式[1]は，参照ポインタにMultiple Reference Bit (MRB) と呼ばれる多重参照情報を持たせて，MRB=OFFのポインタが単一参照であることを保証するようにしたものである．KL1プログラムにおいては，大半のデータは単一参照されているが，MRB=OFFのポインタが消費されればそのデータの占めていたメモリ領域は回収できる．ポインタ側にMRBを持たせたために，回収時以外にはポインタの先にアクセスしなくて済むというメリットがある．これは，クラスタにおいてロックの回数を増やさずに済

＊1 WAMは，現在広く採用されているPrologの抽象マシン語[11]である．

＊2 データのアドレスを動かさないマーク・アンド・スウィープ型のGCは，処理時間やフラグメンテーションの欠陥がある．

＊3 実際には間接的輸出の関係でもう少し複雑な規則となっている[5]．

むことにつながる．MRB情報は，ベクタ要素の更新やストリーム・マージの最適化およびデータ輸出入管理の簡略化にも利用している．

分散処理系では，クラスタ外から参照されているデータの回収が問題となるが，重みつき参照カウント原理を用いたWEC方式[5]によって即時GCを実現した．

MRB方式とWEC方式で回収できないデータがあるため，クラスタ内並列GC[13]および一括型大域GCを検討中である．

5．並列オペレーティングシステムPIMOS

並列オペレーティングシステムPIMOS[2][15]は，並列推論マシン向けの単一ユーザ，マルチタスクのOSである．

オペレーティングシステムの目的を一言で言えば，計算機の持つ物理的な機能をユーザに使いやすい形で提供することである．計算機の物理的機能は大雑把に言えば，計算能力と入出力機能だが，それをユーザに使いやすい形で提供するとは，一つには計算機資源を明快なモデルの形に抽象化することであり，もう一つにはユーザが，故意または過失により，そのモデルに違反したときに，計算機資源および違反していない部分を保護することである．並列オペレーティングシステムPIMOSは，次のような基本方針で設計された．

(1) KL1で記述された純粋な論理型OSであること

これには，OSを記述するレベルで既に計算機資源がユニフォームに抽象化されているというメリット（例えば，メモリ保護の問題が生じない）がある．KL1は外界と相互作用するプロセスの記述に向いており，PIMOSでは計算機資源の多くをKL1プロセスとしてユーザに見せている．入出力デバイスはその典型である*．

ユーザプログラムとOSはオブジェクトとメタの関係にあり，理想的には，OSはユーザプログラムを1レベル下のデータとして制御するべきだが，これは実行効率上の問題があり，現時点では実用的でない．そこで，前述のようにKL1ではFGHCにメタプログラミング機能を組み込んでいる．

(2) 集中型単一OSであること

PIMOSは，プロセッサごとに独立して動作するOSの集合体ではなく，全体として一体であるシステムの並列実行可能な部分を並列に実行する．非定型並列処理においては物理プロセッサを意識しないほうが，自然にプログラムが書けるのである．ただし，PIMが上位層で疎結合マシンであることを考慮して，強い結びつきのあるプロセス群を同じクラスタにマッピングするなどの措置は講じている（PIMOSの論理的構造への影響はない）．

PIMOSの具体的機能としては，タスクの生成・監視・制御，入出力デバイスの管理，シェル機能，などがある．ユーザプログラムは，ソフト例外を上げることによりPIMOSへの要求ストリームを獲得し，このストリームにメッセージを流すことによって必要なPIMOS資源を得るようになっている．すべてのPIMOS資源はその資源固有のプロトコルを持ったストリームとしてユーザに提供される．

メタプログラミング機能を「理想的に」実現していないために，PIMOSではOSをユーザから保護する必要がある．下の例を見よう．ユーザプログラムがN文字を読むためにgetbというコマンドをファイルに送ったとする．

…, Req=[getb(N, Str)|NewReq], …

ここで，Reqはファイルへの要求ストリームである．PIMOSのファイルハンドラ側にはgetbに対応して，次のような節がある．

```
file_handler([getb(N,Str)|NewReq], File)
        :- true |
        readFromFile(File,N,StrRead,NewFile),
        Str=StrRead,
        file_handler(NewReq, NewFile).
```

もしここで，文字列が読み込まれる前にユーザプログラムがStrに例えば整数の0をユニファイしてしまったとすると，ファイルハンドラで読み込まれた文字列StrReadとStrとのユニフィケーションが失敗してしまう．

また，ユーザプログラムが文字数のNを具体化しないまま終了すると，ファイルハンドラ側に中断したままのゴールが残ってしまう．

PIMOSでは保護フィルタというインタフェース・ルーチンを導入することでこれを解決した．保護フィルタは，ユーザがPIMOS資源へのストリームを獲得するたびに，ユーザプログラムとPIMOS資源の間に自動的に挿入されるフィルタ・プロセスであり，次の機能を持つ．

① ユーザプログラムからのメッセージでPIMOS側で読む部分については，それが具体化されるのを待ち，メッセージとして正しい形をしているこ

＊ これに対して，Prologにおいて入出力機能がいかにPrologのセマンティクスと合わない汚い形で入っているかを思い起こされたい．

とを確認した上で，PIMOS側に流す．

② ユーザプログラムからのメッセージで，PIMOS側で書く部分については，新たな未束縛変数に置き換えてPIMOS側に渡し，PIMOS側で具体化した後に元の変数とユニファイする．

保護フィルタの導入によって，ユニフィケーションの失敗やデッドロックはユーザタスク内で起きることになり，PIMOS本体は安全である．また，ユーザタスクの強制終了時にPIMOSへのストリームを閉じるバルブ機構が用意されている．

6. 現状と課題

PIMとPIMOS，およびそれらを結ぶKL1について概説してきた．現在，KL1処理系がマルチPSI*に実装されており，PIMOS暫定版が稼動している．また，1台の小型化PSI上でマルチPSIをシミュレートするPseudoマルチPSIシステムも開発されている．両システムとも徐々にICOT外の研究機関にリリースして並列プログラミング研究に役立てていただく予定である．KL1処理系がPIM/pに実装されるのは来年度半ば頃となろう．

現在のPIMOSでは，入出力機能がすべてFEPにおいて実現されており，コンパイラなどの開発環境もPSI上のクロスシステムに置かれているが，それらの機能をPIMOS本体に移行していくつもりである．

本来OSが管理すべき計算機資源のうち，クラスタごとのメモリ使用量などは，現在KL1の下に隠れてしまっており，きれいな形でそれらを言語レベルに持ち上げて管理の対象にすることは，論理OSとしてのPIMOSの今後の課題の一つであろう．

負荷分散の研究はこれからの分野だが，研究の進展にともない将来的には，実験的な動的負荷分散機能をPIMシステムに組み込んでいきたい．

◇参考文献◇

(1) Chikayama, T. and Kimura, Y. : Multiple reference management in Flat GHC, *Proc. of the Fourth International Conference on Logic Programming,* pp. 276-293 (1987).

(2) Chikayama, T., Sato, H. and Miyazaki, T. : Overview of the parallel inference machine operating system (PIMOS), *Proc. of the International Conference on Fifth Generation Computer Systems 1988.* pp. 230-251.

(3) Goto, A., Sato, M., Nakajima, K., Taki, K. and Matsumoto, A. : Overview of the parallel inference machine (PIM) architecture, *Proc. of the International Conference on Fifth Generation Computer Systems 1988,* pp. 208-229.

(4) Ichiyoshi, N., Miyazaki, T. and Taki, K. : A distributed implementation of Flat GHC on the Multi-PSI, *Proc. of the Fourth International Conference on Logic Programming,* pp. 257-275 (1987).

(5) Ichiyoshi, N., Rokusawa, K., Nakajima, K. and Inamura, Y. : A new external reference management and distributed unification for KL1, *Proc. of the International Conference on Fifth Generation Computer Systems 1988,* pp. 904-913.

(6) Kimura, Y. and Chikayama, T. : An abstract KL1 machine and its instruction set, *Proc. of the 1987 Symposium on Logic Programming,* pp. 468-477 (1987).

(7) Rokusawa, K., Ichiyoshi N., Chikayama, T., and Nakashima, H. : An efficient termination detection and abortion algorithm for distributed processing systems, *Proc. of the 1988 International Conference on Parallel Processing, Vol. I Architecture,* pp. 18-22 (1988).

(8) Sato, M., Shimizu, H., Matsumoto, A., Rokusawa, K. and Goto, A. : KL1 execution model for PIM cluster with shared memory, *Proc. of the Fourth International Conference on Logic Programming,* pp. 338-355 (1987).

(9) Shinogi, T., Kumon, K., Hattori, A., Goto, A., Kimura, Y. and Chikayama, T. : Macro-call instruction for the efficient KL1 implementation on PIM, *Proc. of the International Conference on Fifth Generation Computer Systems 1988,* pp. 953-961.

(10) Ueda, K. : Guarded Horn Clauses ; A Parallel Logic Programming Language with the Concept of a Guard, Technical Report TR-208, ICOT (1986).

(11) Warren, D. H. D. : An abstract Prolog instruction set, Technical Note 309, SRI International (1983).

(12) 沖 廣明，瀧 和男，清 慎一，古市昌一：マルチPSIにおける並列詰め碁プログラムの実現と評価，並列処理シンポジウムJSPP '89論文集，pp. 351-357 (1989).

(13) 佐藤正俊，後藤厚宏：KL1並列処理系の評価—メモリ消費特性とGC—，並列処理シンポジウムJSPP '89論文集，pp. 195-202 (1989).

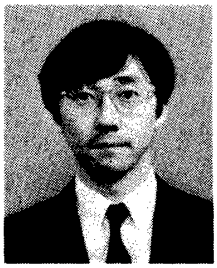
(14) 服部 彰，篠木 剛，久門耕一，後藤厚宏：並列推論マシンPIM/pのアーキテクチャ，並列処理シンポジウムJSPP '89論文集，pp. 107-114 (1989).

(15) 宮崎敏彦：並列論理型言語KL1の実現方式と並列OSの記述，電子情報通信学会論文誌 D，Vol. J 71-D，No. 8，pp. 1423-1432 (1988).

---

* 逐次型推論マシンPSIをメッシュ状のネットワークで結合した疎結合型並列マシン．マルチPSIの要素プロセッサはPIM/pのクラスタに相当すると考えてよい．FEPはシステム全体で最大4台まで接続可能となっている．

著者紹介

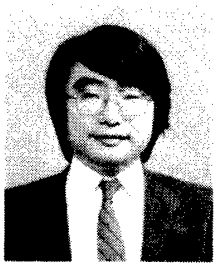


市吉　伸行

1979年東京大学理学部情報科学科卒業. 1981年同大学院修士課程修了. 1982年(株)三菱総合研究所入社. 1984年より1年間米国バテル研究所にてAIを研修. 1987年より(財)新世代コンピュータ技術開発機構へ出向. 論理型言語処理系, 並列プログラミングの研究開発に従事. 情報処理学会, AAAI各会員.

後藤　厚宏

1979年東京大学工学部電子工学科卒業. 1981年同大学院情報工学専門課程修士課程修了. 1984年同博士課程修了. 工学博士. 同年, 日本電信電話公社(現NTT)研究所入社. 1985年より(財)新世代コンピュータ技術開発機構へ出向. 並列推論マシンの研究開発に従事. 電子情報通信学会, 情報処理学会, IEEE, ACM各会員.



近山　隆

1977年東京大学工学部計数工学科卒業. 1979年同大学院情報工学専門課程修士課程修了. 1982年同博士課程修了. 工学博士. 同年, 富士通(株)入社. 同年6月より(財)新世代コンピュータ技術開発機構へ出向. 逐次および並列推論言語と処理系およびオペレーティングシステムの研究開発に従事. 情報処理学会会員.