ICOT Technical Memorandum: TM-0294

TM-0294

人工知能における並列処理

後藤厚宏

April, 1987

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191~5
Telex ICOT J32964

Institute for New Generation Computer Technology

人工知能における並列処理

後藤　厚宏

1. はじめに

並列処理の夢とは何であろうか．並列処理研究の目的の第一は，並列処理がもたらす（と期待される）飛躍的性能の向上，またはコストパフォーマンスの向上である．しかし，実際に並列計算機の開発に携わっている研究者の本音は単なる性能追求だけではない．並列計算機研究の過程において解明されていく計算機処理の本質？ に大きな魅力を感じる．人工知能における並列処理には，未解決の課題が多い．その分，研究者にとって解明する喜びの種にあふれていると言えよう．

人工知能向きマシンを作るということは，特殊な計算機を作ることではない．逆に，これまで“汎用”と呼ばれていた計算機は真に汎用なものではなく，実際にはその応用範囲を絞ることによって計算機処理の価値を高めてきたと考えるべきであろう．計算機の適用範囲を広げるという意味において，人工知能向き計算機の開発は，より汎用な，または，より強力な計算機を作ることと考えてよい．なぜなら，人工知能向き計算機とは，単にゲーム木の探索を行うだけの計算機ではなく，これまでの計算機の持っている多くの機能も兼ね備えていなければならないからである．人工知能向き計算機においても，資源管理やユーザインタフェスを備えるオペレーティングシステム（OS）がさらに重要な意味を持つことは，新世代コンピュータ技術開発機構（ICOT）が逐次型推論マシンPSIとOS（SIMPOS）の開発を通して示した通りである．

人工知能における並列処理を考えると，処理の汎用性/柔軟性をより強く認識する必要がある．例えば，扱うデータ構造はリストのように動的にメモリを使うものがほとんどであり，より強力なメモリ管理やガーベッジコレクションの機能が要求される．また，負荷の大きさが動的に変化する．このため，配列計算のように静的に配置するだけでなく，プロセッサ間の負荷分散を動的に行う機能が必要となる．

このように，人工知能向き並列計算機の研究開発は“計算機システム全体の機能と性能を強化する努力”と考えるべきであろう．

2. 人工知能の並列処理における課題

ここでは，ICOTにおける並列推論システムの研究の過程で明らかになってきた人工知能の並列処理における課題について考えてみる．

2・1　並列推論マシンの核言語とその役割

(1)　マシンからユーザまで一貫した言語系をめざす

ICOTの並列推論システム研究においては，プログラミングからアーキテクチャまでを論理型言語に基づく核言語によって一貫性を持って実現することを一つの目標としている．このような一貫性は言語とアーキテクチャの整合性を高め，性能の向上，システムとしての見通しの良さを実現する重要な条件である．例えば，論理に基づくメタ推論，学習，知識の獲得/管理といった高度な人工知能ソフトウェアまでもが，この一貫したシステムに取り込むことが可能になる．この結

表1

・並列性：ユーザレベルの言語は，従来型の言語に並列処理のプリミティブを追加したような言語ではなく，根本的に並列な言語であり，かつ簡潔であること．
・記述力：並列アルゴリズムを記述できる汎用言語であること．すなわち，同期や通信等の並列処理の基本的な概念が自然に記述できること．
・効率：　効率の良い実現が可能な言語仕様であること．

果，プログラミング技術やコンパイル技術，OS における資源管理手法，機械語からアーキテクチャに至るシステム階層の整合性が確保され，最後はシステム性能の向上に貢献する．

(2)　核言語への要求

並列推論マシン PIM の機械語（ハードウェアとの接点）からユーザ言語までの各レベルをカバーする核言語には，表1に示す並列性，記述力，効率の三つが要求される．

ICOT プロジェクトの初期において，PIM の対象言語として考えられたのは純 Prolog である．しかし，オペレーティングシステム (OS) のように，PIM 上で並列に処理される人工知能処理自体を制御するようなソフトウェアの記述ができるように純 Prolog を拡張することは容易ではなかった．

(3)　Concurrent Prolog から GHC へ

次に，並列に動作するプロセスの制御を記述できる言語として注目されたのが Shapiro 博士が提案した Concurrent Prolog (CP) である．しかし，CP はその言語仕様が複雑であり，効率の良い処理系の実装が難しいという欠点を持っていた．

ICOT において，CP の持つ並列プログラミングの能力と，核言語としての簡明さ，汎用性ある記述能力の双方を持つ言語として生み出された言語が GHC である．

通常，GHC や CP は AND 並列型の論理型言語と呼ばれ，並列プロセスがゴールによって，プロセス間の同期やストリーム通信が AND 関係にあるゴールの共有変数によって表現される．このような並列プログラミングの能力により，人工知能分野の応用プログラムだけでなく，それらの実行を制御する OS 自身も GHC によって統一的に記述することが可能になる．

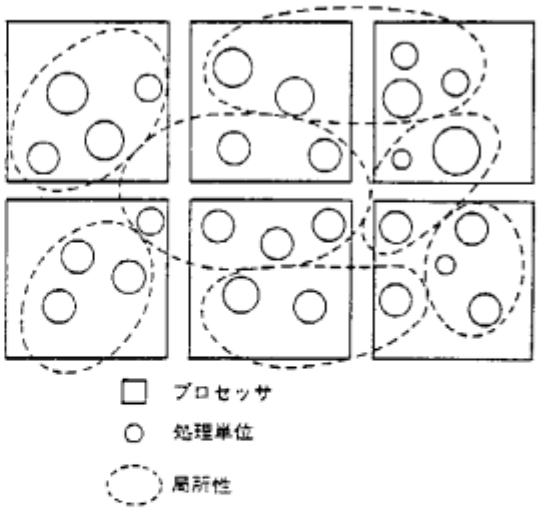


図 1　並列処理の粒度と局所性

2・2　並列推論マシンのアーキテクチャと局所性

並列処理では，常に並列処理することによる効果と並列に処理するためのオーバヘッド（問題を分割したことによって生じる通信のコスト）とのトレードオフを考える必要がある．

並列処理の対象となるプログラムは種々の大きさのアクティビティから構成されており，それには要素プロセッサより小さいものも大きいものもある（図1）．一般的には，前者を並列処理の粒度の問題，後者を並列処理におけるプロセッサ間の通信の局所性の問題として考えられよう．論理型言語で記述されたプログラムであれば，並列に処理するゴールがそのような粒に相当する．

(1)　問題に内在する並列性と並列処理の粒度

通常の逐次型プロセッサにおける並行プロセスの処理を考えてみよう．逐次プロセッサにおいては，プロセスの切換え（コンテキストスイッチ）のコストが高いため，頻繁に切換えが生じると性能が落ちる．しかし，同一コンテキストである限り処理の効率は良い．並列処理における通信のコストはこのようなプロセスの切換えに似ている．つまり，並列処理では一つのアクティビティの処理の効率を保ったままコンテキストスイッチに相当する通信のコストを小さく抑えることが目標となる．

粒度を小さくすることによって並列性は高まる．しかし，実際のプログラムでは，非常に細かい粒の処理

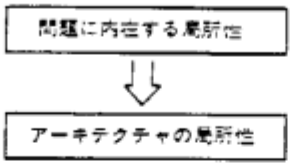


図 2　局所性を対応付ける

単位をそのまま並列処理する必要がない場合が多い．また，細かい粒に問題を分割すると，並列処理の効果に対する通信コストの比率が高くなりがちである．つまり，並列マシン上で考えるべき並列度は，上記の意味で“通信コストが小さく抑えられるような粒の数”として考えるべきである．一方，並列処理の粒度としては，ある程度大きいものを考え，プロセッサがひとまとまりの仕事として実行しやすいサイズのものにまとめる必要がある．

(2)　処理の局所性とアーキテクチャの局所性

問題を構成する処理単位をそれらの間の結び付きの強さ，つまり局所性によって幾つかのグループに分けてみることができる．このような局所性は，並列マシンにおける通信コストの局所性に対応させて考えるべきである．

例えば，データ転送の観点から並列マシンの構成方式を考えた場合，通信のコストの大きい部分と小さい部分が必ずある．例えば，あるプロセッサ内のレジスタからレジスタへの通信は低コストであるが，メモリのある場所から他の場所への通信，さらに他のプロセッサへのメッセージ通信は高コストとなる．

このようなアーキテクチャ上の局所性が存在する以上，ソフトウェアのレベルから，アーキテクチャの局所性を利用すること，および，ソフトウェアから利用し易いハードウェア構成を導入することが必要である．つまり，その間の通信が最小になるように問題を分割し並列マシン内に分散することを目指さなければならない（図2）．このためには，まず，

- アルゴリズムの設計の段階で並列性と局所性を考えること
- プログラミングにおいてそれを明確に表現できること

が大切である．ただし，人工知能処理においては，処理の大きさ（負荷）が常に変化するため，

- 動的に変化する負荷を問題の局所性によって制御できるメカニズム

が必須となる．

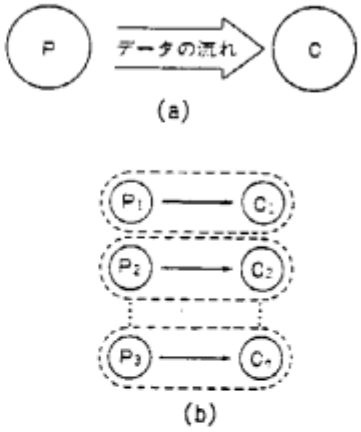


図 3　アルゴリズムにおける通信の局所性

2・3　並列アルゴリズムにおける局所性と負荷分散

処理の局所性を最初に規定できるのはプログラムまたはアルゴリズムである．

並列処理のプログラムにおいては，なんらかの役割を持つ複数のプロセスとそれらの間の通信によって問題を記述する．プログラマにとっては，プロセスの意味が簡明に記述できることが重要である．しかし，並列処理においては局所性についても考える必要がある．

例えば，図3(a)のように，データを生成するプロセスPとそれを消費するプロセスCがあるとする．このようなプロセスの意味は明確で分かり易い．しかし，プロセス間の通信を考えた時，両プロセスは大量のデータを送受し合うかもしれない．このような場合，プロセスPとCを別々のプロセッサに配置するような負荷分散は好ましくない．もし，プロセスPとCが図3(b)のように分割できるようなアルゴリズムの方が局所性を活かした処理が可能になる．

これまで，数値計算の分野では，並列マシンを想定した計算アルゴリズムが検討されてきた．しかし，人工知能の分野においてはほとんどなされていない．人工知能の並列処理を目指す時，並列マシン上での局所性，または並列マシン上での負荷分散を考慮にいれた並列アルゴリズムの検討が大きな課題と言える．

2・4　ハードウェアの役割

人工知能の並列処理においては，先に述べたような

- ソフトウェアから利用し易いアーキテクチャ上

の局所性

● 人工知能処理に適したマシン命令セット

がハードウェア設計における中心課題である．

さらに，システムにおけるソフトウェアとハードウェアの役割分担を考え直す点もある．ハードウェアの役割は，"処理全体において比較的単純であるにもかかわらずその頻度が多いため性能を抑えている要因" を解決することであろう．この意味では，人工知能の並列処理においてハードウェアが分担すべき項目として，

① タグによるデータ型判定に伴う処理

② アドレス変換

③ ガーベッジコレクション

等があげられる．①については，これまでに開発された逐次型の Lisp マシン，Prolog マシンを通した技術の蓄積がある．一方，②，③については，新たに考えなければならない点が多い．

例えば，並列マシン，特にメッセージ通信に基づく疎結合システムにおいては，従来の論理/物理アドレス変換だけでなく，プロセッサ間でのメッセージ通信にかかわるアドレス変換が加わる．このため，②アドレス変換の機構のハードウェア化はこれまで以上に大切となろう．また，人工知能処理におけるガーベッジコレクションは，プログラムの実行と同等に扱うべき重要な処理である．ガーベッジコレクションをプログラムを実行する命令の機能の中に埋め込むような機構も必要となろう．

3. 並列推論システムの開発

並列推論システムの研究は，

● 対象とする応用分野/応用問題

● 処理アルゴリズム

と協調しあって進まなければならないものはもちろんのこと，並列処理システムを構成する，言語，処理方式，アーキテクチャを一貫性を持って作り上げる必要がある．

ICOT では，論理型の枠組みの基に，

① ユーザ側からマシン側までを一貫性を持ってサポートする論理型言語システム（KL 1），

② KL 1 で記述した並列推論システムとしての機能

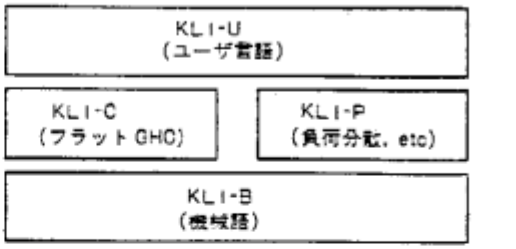


図 4　核言語 KL 1 の言語系

を持つオペレーティングシステム（PIMOS），

③ 並列推論マシン（PIM）アーキテクチャ，

の三つを統一的に開発することを目指している．さらに，並列推論システムにおけるソフトウェアの蓄積を，

④ 並列ソフトウェアの開発環境（マルチ PSI），

によって支援する．

3-1　核言語 KL 1 システム

核言語 KL 1 は，GHC をベースにユーザ側，マシン側の双方に拡張した階層を持つ言語系として構成する（図 4）．

(1) KL1-c (ore)

KL 1-c は，GHC の機能を一部制限した Flat GHC にメタ機能を追加した言語であり，まさに KL 1 の核としての役割を持つ．つまり，後述する KL 1 の上位言語 KL 1-u での言語上の拡張の意味はすべて KL 1-c で規定される．また，下位の言語（機械語）KL 1-b は，KL 1-c の実現手段として位置付けられる．

(2) KL 1-p (ragma)

KL 1-p は，先に述べたようなプログラミングにおける局所性や負荷分散の表現のために導入する言語機能である．

プログラマは，KL 1-p の機能を用いて，自分のプログラムが PIM 上でどのように負荷分散した方が良いかについてのヒントを PIM 上の処理系とその OS に与える．このため，KL 1-p は，KL 1 言語系において PIM のアーキテクチャ上の特性が反映される部分と考えてよい．

(3) KL 1-u (ser)

KL 1-u は，KL 1 のユーザ側のインタフェースであり，実際上は，オペレーティングシステム (PIMOS) の記述言語としての役割を持つ．

PIMOS 記述のために必要な KL 1-u の機能としては，

- メタプログラミング機能
- 速度向上のための機能
- 大規模プログラミングのためのモジュール化機能

が不可欠となる．

(4) KL 1-b (ase)

KL 1-b は，KL 1 のマシン側のインタフェースであり，次に述べる PIM の機械語として位置付けられる．

3・2 核言語の処理方式と機械語

核言語 KL 1 の処理モデルは並列ゴールリダクションとしてとらえることが自然であろう（図 5）．

並列ゴールプール内のゴールはゴールスケジューラによって選択されると受動部（頭部とガード）がテストされ，コミットされた場合はその能動部に実行が移る．その結果生成された新ゴールは並列ゴール間の論理関係を管理する構造に登録される．

変数の具体化待合せのために，受動部のテストで中断してコンテキストスイッチする場合と，能動部の実行によって中断ゴールが再開する場合がある．

このような KL 1 の機械語設計におけるポイントには次のものが上げられる．

(1) ゴールの実行制御

システム内に存在する多数のゴールは，概念上並列に処理可能である．ただし，すべてのゴールがいつでも実行できるわけではない．それをリダクションするために必要な入力引数がプログラムの定義節として指定されているためである．つまり，KL 1 のゴールを実行するためには，データフローマシンのように，ゴールの引数の待合せを実行時に制御する機構が必要である．ただし，多くのゴールはすでに引数がそろっていることが多いため，マッチングストアのようなハードウェアを用意せず，スケジューリングキューと Lisp におけるバインドフック機構に相当するデータ構造によってゴールの同期制御を実現する．

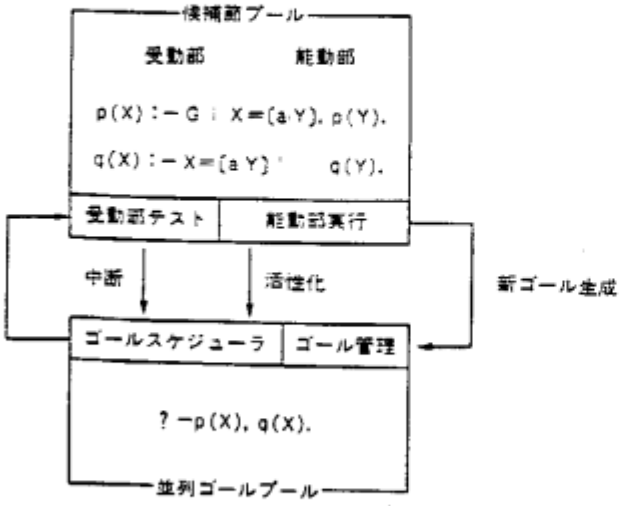


図 5 KL 1 プログラムのゴールリダクション

(2) ユニフィケーションとコンパイル

KL 1 におけるユニフィケーションの大部分は，コンパイラによって，引数の読出し，データ型の判定，メモリへの値の書込みといった基本操作に分解できる．さらに，コンパイル時にゴール間の依存性のデータフロー解析を行い，引数を待ち合わせるゴールを早期に検出できる場合も多い．

このため，コンパイル技術と機械語設計は並列推論マシンにおいても性能の鍵を握るといってよい．

(3) ストリームプログラミング支援

KL 1 においては，ゴールによって表現されたプロセス間をストリームによって接続するようなプログラミングが基本形と考えてよい．ストリームは一般のリスト構造によって実現できるが，ストリームの併合/分割までを考えるとけっして効率が良くない．このため，ストリームプログラミングを効率良く実行できる表現法(CDR-Coding)とその操作プリミティブを機械語のレベルでサポートすることがポイントの一つとなる．

3・3 オペレーティングシステム：PIMOS

PIMOS は，KL 1 で最初に記述される最も規模の大きい並列プログラムである．論理型プログラミングの機能に基づいて，できる限り超論理的な機能を用いずに記述することが並列に動作させる上で重要である．KL 1 は，刻々と変化する並列計算過程を表現する能力に優れており，超論理的な機能をほとんど用いずに OS の機能を記述することが可能であろう．

PIMOS は，集中型の OS であり，いわゆる分散型 OS ではない．KL 1 で記述されたソフトウェアに対して PIM を 1 台の計算機として扱う．このため，人工知能の並列処理としては，PIM の資源管理とそれに基づく負荷分散が PIMOS において最も重要な機能の一つといえよう．PIM においては，通常の計算機の資源が要素プロセッサの台数分だけあることに加えて，それ

それの資源の関係が単純ではない．そこで，ユーザのゴールのかたまりごとにCPU時間やメモリ量等の資源を与えられるようなメタルレベルのモジュールを導入する．PIM上での動的な負荷分散は，PIMOSが提供する資源の利用情報と，プログラム中に示されたプログラマからの指示の双方を用いて実現される．

さらに，本格的なOSとして実用に耐えるために必要な機能を網羅すること，およびソフトウェア開発にとっての使い勝手の良さが重要である．

3･4 並列ソフトウェア開発用マシン：マルチPSIシステム

並列アルゴリズムやPIMOSの開発には，従来型の計算機上でのクロスシステムだけでなく，実際の並列マシン上でプログラミングを繰り返していくことが重要である．ICOTでは，そのような並列ソフトウェアの開発環境として，ICOTで開発した逐次型推論マシンPSIをネットワークで結合したマルチPSIシステムを開発し，その上でプログラムの積重ねを進めている．

(1) マルチPSI（第1版）

実用に耐える並列ソフトウェア開発環境を作ることは，見方によってはPIMそのものを作ることに等しい．マルチPSI第1版は，開発環境としてというよりも，そのパイロットシステムとして開発したシステムであり，KL1の並列処理系とプログラム特性の解析を進めている．

マルチPSI第1版は，ICOTで開発した逐次型推論マシンPSIを格子上のネットワークにより6〜8台接続した構成であり，PSIのユーザ言語ESPを用いてKL1プロトタイプの並列処理系が記述されている．

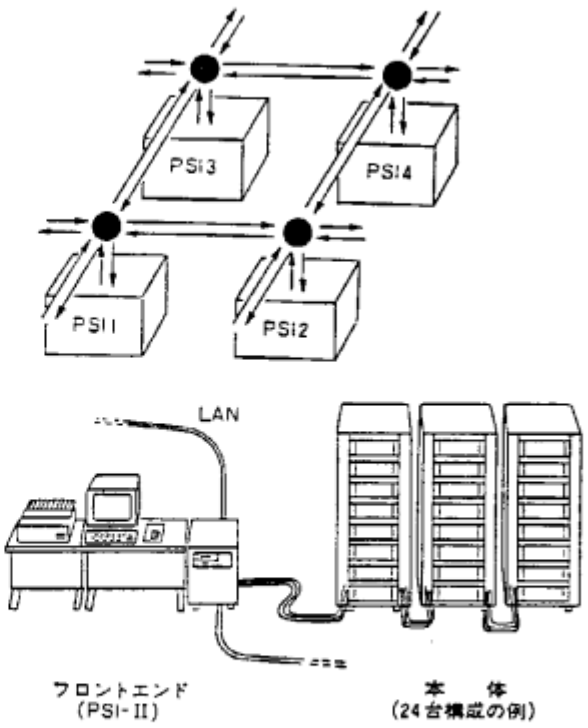


図 6　マルチPSI(第2版)

(2) マルチPSI（第2版）

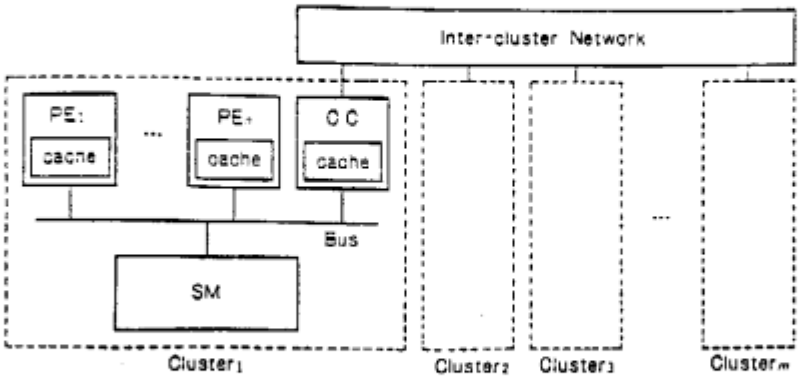
マルチPSI第2版は，本来の目的である並列ソフトウェアの開発環境として設計されたシステムであり，昭和62年度中に稼働する予定である．

要素プロセッサとして小型化/高速化したPSIを新たに開発したため，システムをコンパクトにしたまま最大64台の要素プロセッサを接続できる（図6）．

KL1の並列処理系は，要素プロセッサのファームウェアによって実装し，マルチPSI第1版の数百倍の性能を目指している．

(3) マルチPSIの処理方式

マルチPSIの各プロセッサには，メッセージ通信の



PE：Processor Element
SM：Shared Memory
CC：Cluster Controller

図 7　並列推論マシンPIMのアーキテクチャ

ためのプリミティブがファームウェアで用意されており，これを用いて可変長のメッセージを任意のプロセッサに対して送受信できる．各プロセッサで動作するスケジューラは，メッセージとしてユーザゴールを別のプロセッサに分散する．プロセッサ間の通信は，他プロセッサ上のゴールとの間の共有変数へ値をユニフィケーションすることを契機に処理系が自動的に行う．このため，ユーザが陽にプロセッサ間通信を記述する必要はなく，あくまでソフトウェア上で通信しあう並列プロセスと，それらのプロセスをどのように各プロセッサに配置するかをKL 1-pの機能によって指定すればよい．

3・5　並列推論マシン PIM

(1) 開　発　目　標

ICOTで開発を進めているPIMでは，核言語KL 1を高速実行と共に，並列OS（PIMOS）が安定して稼働できる実用規模/実用性能を有するマシンを目指している．

要素プロセッサは100台規模で接続され，その目標性能を要素プロセッサ当り200〜500 KLIPS，システムで10〜20 MLIPSにおいている．

(2) PIMのアーキテクチャ

アーキテクチャ上での局所性に基づくPIMの構成を図7に示す．

要素プロセッサ(PE)は，KL 1用に新規に設計することにより，単体でも現在の高性能計算機と同等以上の性能を持つものを目指している．また，小型化/高集積化することによりプロセッサ間の通信コストを下げ，全体性能を上げることを目指す．

8〜10台程度のPEは共有メモリと共有バスを介して接続され一つのクラスタを構成する．クラスタは，いわゆる密結合マルチプロセッサであり，共有メモリのアドレス空間を共有しあうことにより，PEどうしが通信する．各PEは共有メモリへのアクセスを高速化するためのキャッシュメモリを持つ．このキャッシュメモリには，共有メモリの内容のコピーを正しく維持するための機構が設けられる．

一方，各クラスタはメッセージ通信ネットワークを用いた一種の疎結合マルチプロセッサを構成する．各クラスタには，クラスタ間におけるメッセージ通信を強化するためのクラスタコントローラ（CC）を設け，メッセージ通信に必要なクラスタでのアドレス変換を高速化する．

(3) プロセッサ間通信

PIMにおけるプロセッサ間通信には，負荷(ゴール)分散とユニフィケーションに伴う遠隔データアクセスやプロセッサ間にまたがるゴールの同期制御がある．

PIMのクラスタ内においては，プロセッサな共有メモリ上の変数を介して通信し合う．このため，メモリアクセスの排他制御機構が重要である．PIMにおいては，各プロセッサのキャッシュのブロック状態を利用したロック機構を設け，低コストの排他制御を可能にする．

一方，クラスタ間にわたるプロセッサ間通信はマルチPSIと同様にメッセージのパケット通信によって行う．ここで，ネットワークのトポロジー以上に重要なのが，メッセージ通信におけるパケットの組立て/読取りを支援するハードウェアとクラスタ間にわたるポインタの管理である．特に，クラスタ間にわたるポインタの管理は，アドレス変換に相当する操作であると同時に，システム全体のガーベッジコレクションの実現方法と強く関連する．

4. おわりに

ICOTにおける並列推論システムの研究開発において明らかになってきた人工知能における並列処理の課題と，現在ICOTで進めている研究内容について述べてきた．

人工知能に限らず，並列処理にはいまだ多くの課題があるばかりでなく，今後の研究が進むことによって明らかになる課題も多数あろう．これは，関係する者にとっては“やりがい”のかたまりである．