TR-0869

LSI配線プログラムを用いた並列推論マシン PIM/c の負荷分散方式の評価

朝家　真知子、中川　貴之、
垂井　俊明、井門　徳安、
杉江　衛（日立）

April, 1994

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03)3456-3191～5

Institute for New Generation Computer Technology

LSI配線プログラムを用いた並列推論マシンPIM/cの負荷分散方式の評価

朝家真知子†、中川貴之‡、垂井俊明†、井門徳安†、杉江 衛†
†（株）日立製作所　中央研究所
‡（株）日立製作所　汎用コンピュータ事業部

概　要

主記憶共有のプロセッサエレメントをネットワーク結合した階層構成型並列推論マシンPIM/c（256プロセッサ構成）上にLSI配線プログラムを実装し、システム性能を評価した。クラスタ構成に適したプロセス割当方法により、960ネット配線問題において256プロセッサで94.8倍の台数効果を達成した。また、動的負荷分散支援ハードウエアを活用し、ソフトウエア動的負荷分散制御に比べて1.8倍に高速化した。

Abstract

Parallel Inference Machine, PIM/c, 256-processor prototype has hierachical cluster structure. We evaluated the system performance of PIM/c, using parallelized LSI routing program. With the new process mapping strategy for PIM/c hierachical system, 256-processor sysytem can execute 95 times faster than one-processor system.

Furthermore, it is confirmed that utilizating hardware to support dynamic load balancing the routing program can be executed 1.8 times faster than software-controlled dynamic load balancing.

1．はじめに

（財）新世代コンピュータ技術開発機構（ICOT）では、通産省の第五世代コンピュータプロジェクト（昭和５７年度～平成４年度）の一環として、大規模知識処理を目指した、並列推論マシンの研究開発を進めてきた[1]。我々は、ICOT再委託研究として、256プロセッサで構成する並列推論マシンPIM/c (Parallel Inference Machine model c)[2]を開発した。PIM/cは、8プロセッサエレメントでメモリを共有するクラスタを32台接続し、クラスタ間はメモリを分散する階層構造を採用している。

我々は、ICOTで開発された並列知識処理応用プログラム（ICOT無償公開ソフトウェア[3]）の中から並列化版「LSI配線プログラム[4]」を選択し、PIM/c上に実装してシステム評価を行なった。「LSI配線プログラム」は、LSIチップ上の部品間を配線するプログラムであり、オブジェクト指向プログラミング手法を用いて配線並列化を行なっている。このプログラムは、1ノード1プロセッサ構成の階層構造を持たないフラットな分散メモリ型並列計算機向きに開発された。並列計算機を効率良く実行させるには、仕事（負荷）を各プロセッサに均等に分配し、プロセッサ間の通信を極力削減することが必須である。1ノード(クラスタ)を複数のプロセッサで構成する階層構造のPIM/cに「LSI配線プログラム」を移植するにあたっては、1ノード1プロセッサ構成計算機用に「LSI配線プログラム」の設計段階で検討された以上に、ノード(クラスタ)間通信量を削減し、ノード(クラスタ)内通信の局所性を生かした静的負荷割当ての検討が必要となった。

また、推論問題には、推論過程の途中で動的に次に実行すべき処理を選ぶという特徴がある。そのため、プログラム実行前にあらかじめ負荷量を推測し、固定的な負荷割当てを行うことは困難である。動的に負荷量が変化する場合、各ノードの負荷バランスを判断して均等に負荷割当てを行な

Evaluation of Load Balancing Strategy using LSI Routing Program on Parallel Inference Machine PIM/c

Machiko ASAIE†, Takayuki NAKAGAWA‡, Toshiaki TARUI†, Noriyasu IDO†, Mamoru SUGIE†
†Central Reserch Laboratory,Hitachi,Ltd.
‡General Purpose Computer Division,Hitachi,Ltd.

うことを動的負荷分散と呼ぶ。動的負荷分散をソフトウエア制御するためには、各ノードに負荷量を問い合わせ、処理待ち状態のノードに明示的に割り当てる必要がある。PIM/cでは動的負荷分散を高速に行なうため、ノードの負荷量を高速に取得し、負荷分散先ノードを判断するためのハードウエア支援機構[5]を搭載している。本稿では、その負荷分散支援ハードウエア支援機構の実用性を検証する。

本稿では我々の開発した並列推論マシンPIM/c上で並列化版「LSI配線プログラム」を動作させて、次の項目を評価する。

(1) クラスタ構成向け静的負荷割当て方式
(2) PIM/cの動的負荷分散ハードウエア支援機構

2. 並列推論マシンPIM/cの構成

2.1 プロセッサ構成

図1に、PIM/cのプロセッサ構成を示す。プロセッサ・エレメント(PE)8台がスヌーピングキャッシュを介して主記憶を共有し、1クラスタ(CL)を構成する階層構成を採用している。複数のプロセッサ間の通信のために、クラスタに1つのクラスタ・コントローラ(CC)を設け、クラスタ内の8プロセッサとクラスタ外のプロセッサとの通信メッセージを一括して制御している。クラスタ・コントローラをクラスタに1つずつ設けることにより、プロセッサがそれぞれ独立に通信路を設けてクラスタ外のプロセッサと直接交信する場合に比べて、プロセッサ間ネットワークのハードウェア量を1／8に抑えている。PIM/cシステムは、クラスタ32台をクロスバ・ネットワークで接続して、全256プロセッサで構成している。

2.2 負荷分散方法

PIM/cにおいて、クラスタ内とクラスタ間で異なった負荷分散方法を使用する。クラスタ内の主記憶共有プロセッサにおいては言語処理系による自動負荷分散、クラスタ間においてはユーザがプログラム中に負荷分散先を指定する述語を記述する明示的な負荷分散を行なう。

階層構成のため、全プロセッサを効率良く稼働させるには、各クラスタ内処理に対しても、8プロセッサが十分稼働できる並列性を持たせることが必須である。また、クラスタ・コントローラが8プロセッサの通信処理を一括制御するため、クラスタ・コントローラの処理が過大とならないようにクラスタ間処理を最小限にとどめるプログラミング手法が必要となる。

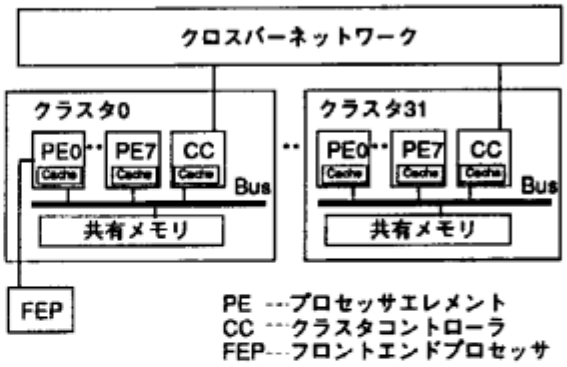

図1. PIM/cのプロセッサ構成

2.3 動的負荷分散支援ハードウェア機構

PIM/cではクラスタ間における動的負荷分散を効率良く実現するために、クラスタ負荷量問い合わせを高速化するハードウエア機構を備えている。動的負荷分散では、クラスタ負荷値を比較し、より小さい負荷値を持つクラスタに処理を依頼する。PIM/cにおける動的自動負荷分散方式では、負荷分散先クラスタはスマートランダム方式[6]によって選択する。スマートランダム方式とは、負荷を分配しようとするクラスタが、ランダムに選択した1クラスタと自クラスタの負荷値比較を行い、相手側の負荷値が小さい場合にのみ負荷を分配し、相手側の負荷値が大きい場合は分配しない方式である。負荷分散時に1クラスタとのみ負荷値比較を行なうので、全クラスタに負荷値を問い合わせるための通信集中が生じない利点がある。以下に述べる、通信高速化ハードウェアを用いて負荷値レジスタの比較によって負荷分散先を決定する。

図2にPIM/cで実現した通信高速化機構を示す。クラスタ間の処理では共有メモリがないので、大域的な変数であるクラスタ負荷値を参照する場合には、他クラスタにメッセージを発行して負荷値を問い合わせ、結果のメッセージを待つ必要がある。通常メッセージ(1)はクラスタとネットワークの間で一旦バッファリングされ、相手先のクラスタがビジー状態であれば、それ以前のメッセージが全て処理されるまで待ち合わせる必要がある。PIM/cでは、負荷値通信の際のこのようなバッファの待時間を削減するために緊急メッセー

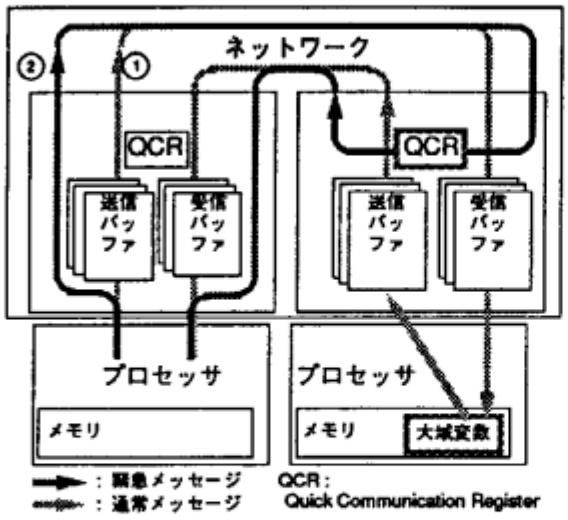

図2．通信高速化機構

ジ専用の短絡路(2)を設けた。さらに、クラスタの負荷値をネットワーク中の簡易レジスタQCR (Quick Communication Register) に格納することで、クラスタ間の負荷値報告の通信処理を高速化している。プログラム中の記述方法は、負荷分散指定箇所に1語の述語を追加するだけで、処理系が自動的に負荷分散先クラスタを選択することができる仕様とした。

3．LSI配線プログラム概要

ICOTで開発されたLSI配線プログラムは、LSI配置図上の部品の端子間を配線するプログラム（ソース量KL1言語5000行）である。縦方向・横方向の格子で配線層を使い分ける2層配線を扱う。回路情報として、格子サイズ・配線禁止領域・スルーホール禁止点・ネットリストの情報を与えて、配線を計算する。

このプログラムの並列化手法は、並列オブジェクトモデルに基づく。図3にプロセス構成を示す[4]。配線格子上における全ての配線線分をオブジェクト＝プロセスに対応させ、これをラインプロセスと呼ぶ。縦横の格子の端から端までの線分をマスタラインプロセスと呼ぶ。マスタラインプロセスは並列に配線探索し、それぞれ直行するラインプロセスとメッセージを交換して、ターゲット点に最も近い位置（期待位置）を返答したラインプロセスを選択して配線を決定する。LSI配線プログラムは、このマスタラインプロセス単位に実行クラスタに配置し、並列計算機にマッピングしている。複数配線処理の同時並列実行、および、複数のラインプロセスに同時に問い合わせを行なうことによる期待位置計算の並列実行の二種類の方法で並列処理を実施している。また、配線領域は早いもの勝ちで確保するため、後に同一配線領域上に配線しようとするプロセスは失敗する。

図4にプロセスとメッセージの流れを示す[7]。マスタライン・プロセスおよびライン・プロセスの他に、ディストリビュータ・プロセスを各クラスタに1個配置し、他クラスタ上のマスタライン・プロセス、ライン・プロセスとの通信要求を一括処理して、各クラスタ間のメッセージ数を最小化している。

4．PIM/cにおけるLSI配線プログラムの負荷分散方式

4.1　プロセスの静的割当て方式

LSI配線プログラムについて、PIM/c上の負荷の静的クラスタ割当て方法変更を検討して、性能向上を図った。

ICOT開発のオリジナルLSI配線プログラムは、階層構造を持たないフラットな分散メモリ型並列計算機上で開発され、各ノードの負荷の均等化を重点課題として設計された。負荷均等化を実現するため、配線図の縦横の格子（グリッド）上のマスタライン・プロセスを順番に（サイクリックに）ノードに割当てている。これを負荷均等化マッピングと呼ぶことにする。図5にプロセスの負荷均等化マッピング方法を示す。このプログラムをPIM/cに実装すると、負荷均等化マッピングにより各クラスタの稼働率は均等化する利点がある反面、図5中の(1)から(2)を結ぶ短い配線でも、複数のクラスタの間に配線処理が生じる欠点がある。

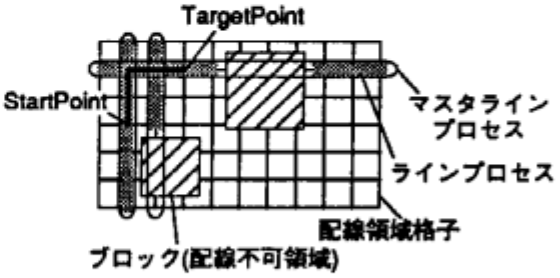

図3．プロセス構造

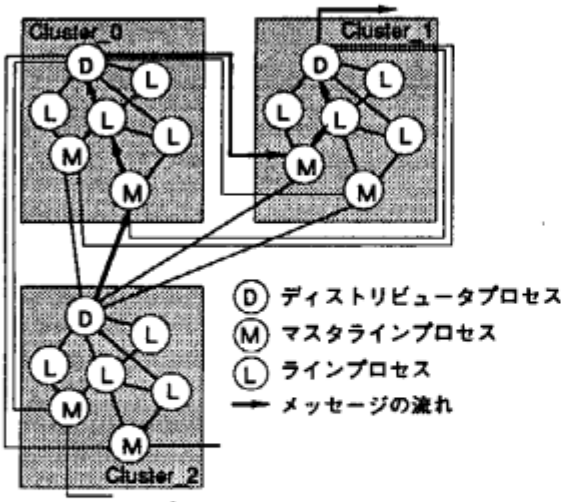

図4．プロセスとメッセージの流れ

PIM/cのプロセッサ構成を考えると、クラスタ内はメモリを共有しているのでクラスタ内処理は高速に扱うことができ、

クラスタ内プロセッサ間処理時間

＜　クラスタ間処理時間

の式が成り立つ。一般に、LSI配線の問題を扱うとき、配線しなければならない端子間の距離が遠く離れていることは考えにくく、局所的な配線が多いことが予想できる。従って、配線データの局所性を活かしたプロセス割当てを行い、クラスタ内プロセッサ間処理を増やすことによって、クラスタ内処理とクラスタ間処理のレイテンシ差を利用したプログラムの実行高速化を図ることが出来る。

そこで、クラスタ内プロセッサ間処理を増やし、ネットワーク経由のクラスタ間通信を減らすために、図6のようにプロセスの通信局所化マッピングによるプロセス割当て方式を検討した。本方式では、縦・横の格子数、すなわちマスタライン・プロセス数を実行可能な最大クラスタ数に分割して、複数のマスタライン・プロセスの単位で連続的に割当てる。その結果、図6の例で示す配線では始点(1)と終点(2)が同じクラスタCL0内に存在する。

さらに、通信を局所化するマッピング方式を採用することによって、クラスタ間に発生した無駄な配線処理を削減する効果もある。並列処理の過程では、必ずしも最短経路のみを探索するわけではないため、配線領域が空いていれば、複数のプロセスが探索を行う。複数の配線が同一の配線領域を確保しようとするときには、最初に配線終了メッセージを受理されたプロセスだけが選ばれ、他のプロセスは失敗する。したがって、クラスタ内のマスタライン・プロセスに始点・終点双方が存在する場合に、クラスタ内通信処理がクラスタ間通信処理に比べ高速なことによって、通信レイテンシ差によりクラスタ内のプロセス間で配線処理を終了することができ、クラスタ間に関与してしまうような配線処理はキャンセルできる。

4.2　動的自動負荷分散支援ハードウェアを用いた負荷分散方式

動的な負荷分散の観点からも、PIM/cの性能評価を試みた。以下の条件で、ソフトウェアで動的負荷分散制御を行なった場合と、PIM/cの独自機構である動的負荷分散ハード支援機構を使った自動負荷分散方式について、LSI配線プログラムに応用して実験および性能比較した。

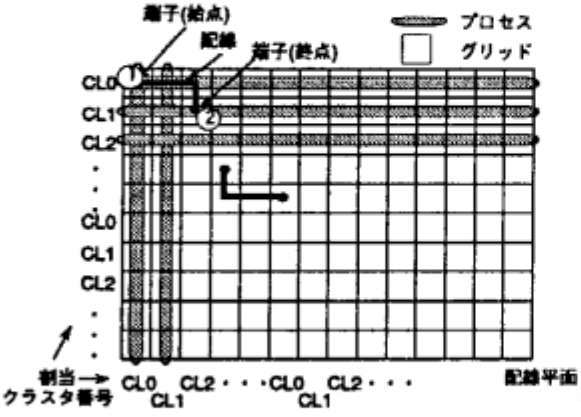

図5．プロセスの負荷均等化マッピング方法

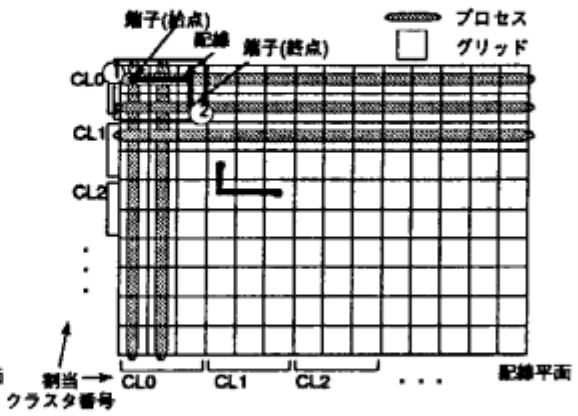

図6．プロセスの通信局所化マッピング方法

(1) ソフトウェア制御による動的負荷分散における負荷分散先選定方法

プログラム実行中に負荷分散をする時点で、負荷分散元クラスタが全クラスタに問い合わせ通信を行ない、処理待ちプロセスを持たないクラスタに対して、負荷分散する。

(2) ハードウェア制御による動的負荷分散における負荷分散先選定方法

処理待ちプロセス数を負荷値と定義し、負荷値レジスタ(動的負荷分散ハード支援機構のQuickCommunicationRegister)の値をクラスタ間で比較、スマートランダム方式で選択したクラスタに負荷分散する。

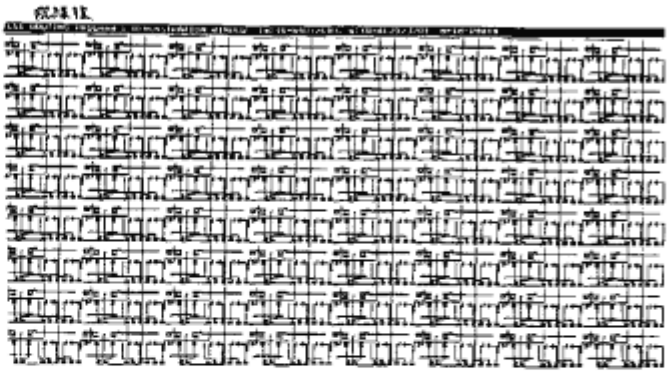

図7. LSI部品配置図(960NETS)

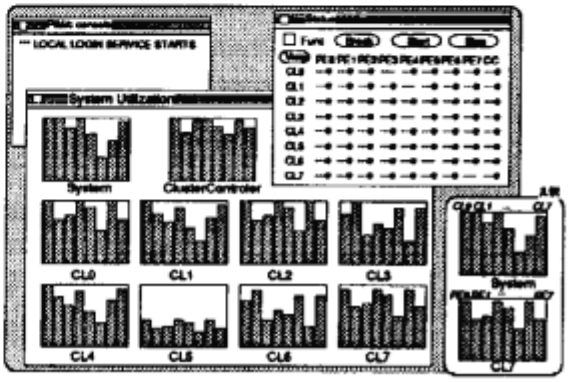

図8. SVP稼働率モニタ

5. 実験方法および評価結果

5.1 実験方法

5.1.1 プロセスの静的割当て方式

「4.1 プロセスの静的割当て方式」で述べた方法に従って、

(1) 負荷均等化マッピングの場合(図5)

(2) 通信局所化マッピングの場合(図6)

について、性能を計測した。

図7に示す配線数960ネット（格子数640×320）データを使用して、プロセッサ台数1,8,16,32,64,128,256台について実行時間を計測し、台数効果を算出した。

負荷バランスの観察および確認には、SVP稼働率モニタ図8を利用した。PIM/cのサービスプロセッサ上に実現した稼働率モニタで、言語処理系上で実行したアイドルゴール数をカウントし、(全サイクル数-アイドルゴール実行数)をプロセッサ稼働率としてリアルタイムに表示している。図9、および図10は、32クラスタ（256プロセッサ）でLSI配線プログラムを実行したときの3次元稼働率グラフを示し、SVP負荷バランスモニタの時間による変化を表している。load軸は稼働率、clusters軸はクラスタ番号、time軸は稼働率計測時間軸を表す。図9は各クラスタにおける8プロセッサの稼働率平均値を、図10はクラスタ間処理を行なうクラスタ・コントローラの稼働率を表す。time20～time40の部分が配線処理におけるプロセッサ稼働率を表している。クラスタ内処理はほぼ100%、クラスタ間処理（クラスタ・コントローラの稼働率）は50%程度で稼働し、クラス

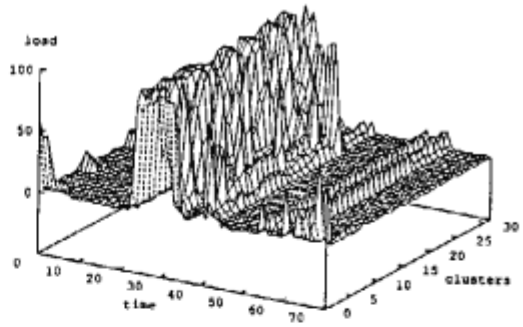

図9. プロセッサの稼働状況
（クラスタ内処理）

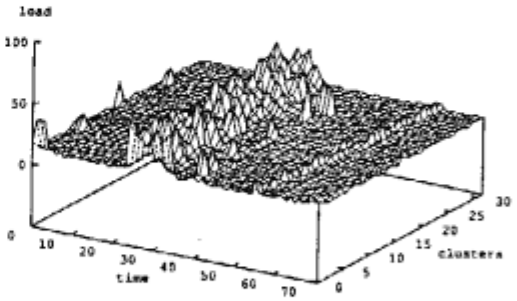

図10. プロセッサの稼働状況
（クラスタコントローラ処理）

タ·コントローラの通信処理が全体のボトルネックにならない理想的な稼働状態の下で計測した。

また、配線問題の実行時間計測直前にGCガベージコレクションを実行し、配線実行中のGCによる誤差を回避して計測した。

さらに、(1)および(2)方式のクラスタ間処理を比較するため、図4に示す各クラスタのディストリビュータ·プロセスが他のクラスタに存在するマスタライン·プロセスへ転送するメッセージ種類·数を、136ネットデータを利用し、8クラスタで実行した場合について測定した。

5.1.2 動的負荷分散

プログラムの実行負荷が動的に変化する箇所を選択し、以下の2方式について、プログラムに動的負荷分散表現を追加して、実行時間およびプログラム表現ステップ数を比較した。配線データは136ネット規模データ、実行クラスタ数は8クラスタ（64プロセッサ）で比較した。

図11、12にプログラム比較を示す。

(1) ソフトウェア制御動的負荷分散（図11）

実行すべき処理を持たないクラスタを検出して処理依頼する。本方式では、述語 get_hima_node により全クラスタにプライオリティ最低値の簡単な処理を依頼し、実行結果を最初に応答したクラスタを実行すべき処理を持たないクラスタと判別、負荷分散先に選

```
                    ...
line([Message|Ms], LineData) :-
    vector_element(Message, 0, Name), Name = draw_line |
      get_hima_node(Node),
      draw_line([Message|Ms], LineData) @node(Node). % dynamic_distribution.
line([Message|Ms], LineData) :-
    vector_element(Message, 0, Name), Name = route |
      get_hima_node(Node),
        route([Message|Ms], LineData) @node(Node). % dynamic_distribution
                    ...
%%%
get_hima_node(HNode):- true |
      merge(In,Out),
      himakai_node(0,8,In)@priority($,4096),
      himada_node(Out,HNode).

himada_node([Node|_],HNode):- integer(Node) | HNode = Node.

himakai_node(C,T,M):- C = T |
      himakai_node(~(C+1),T,M).
himakai_node(C,T,M):- C >= T | M = [].
otherwise.
himakai_node(C,T,M):- C < T |
      himakai(C,M1)@node(C),
      M = {M1,M2},
      himakai_node(~(C+1),T,M2).

himakai(C,W):- true | himada(C,W) @priority($,-4096).
himada(C,W):- integer(C) |  W = [C].
                    ...
```

図11. ソフトウェア制御動的負荷分散記述方法

```
                    ...
line([Message|Ms], LineData) :-
    vector_element(Message, 0, Name), Name = draw_line |
        draw_line([Message|Ms], LineData) @node(-1). % dynamic_distribution.
line([Message|Ms], LineData) :-
    vector_element(Message, 0, Name), Name = route |
        route([Message|Ms], LineData) @node(-1). % dynamic_distribution
                    ...
```

図12. ハードウェア制御動的負荷分散記述方法

択して負荷分散指定子@nodeにより処理を分配する。

(2) ハードウェア制御動的負荷分散 (図12)

動的自動負荷分散指定子@node(-1)を依頼したい処理に付加し、スマートランダム方式で依頼先クラスタを選択し、処理を分配する。

5．2　評価結果

5.2.1　プロセスの静的割当て方式

配線数960ネット(格子数640×320)データを使用して、プロセッサ台数1,8,16,32,64,128,256台について、実行時間を計測した結果、ほぼ97%程度以上の配線率を得た。台数効果の結果を図13に示す。256PE構成の1PEに対する相対性能は、負荷均等化マッピングの場合に82.3倍、通信局所化マッピングの場合に94.8倍である。また、256PE構成の実行時の負荷均等化マッピングと通信局所化マッピングの方式を比較した結果、実行時間にして95秒、および81秒であり、プロセスの割当てを変更し通信局所化マッピングした結果、約1.2倍に性能を向上させることに成功した。

各クラスタ内の処理内容は、ほとんどが配線を探索している部分で100%近いプロセッサ稼働率が観察できた。その間のクラスタ・コントローラの稼働率は、50%程度である。クラスタ・コントローラの通信処理が全体のボトルネックにならずに、クラスタ内の処理が十分行なえており、並列処理の理想的な稼働状況を示している。

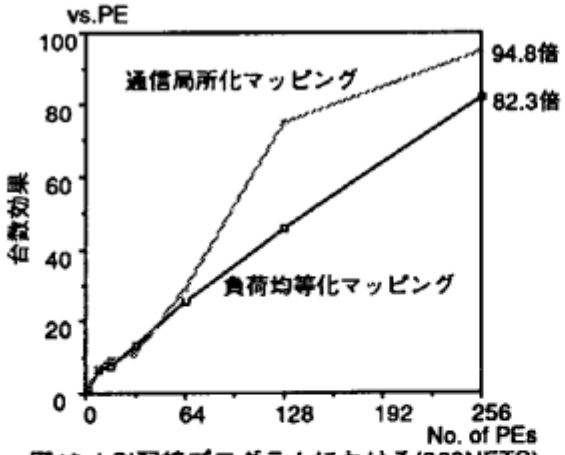

図13. LSI配線プログラムにおける(960NETS)台数効果

クラスタ間処理を示すディストリビュータ・メッセージ数と内訳を表1に示す。send_master2メッセージとは、図4においてディストリビュータ・プロセスが他クラスタのマスタライン・プロセスに発行するメッセージである。136ネットデータを利用した計測では、負荷均等化マッピングの場合に比べて　信局所化マッピングの場合、クラスタ間通信メッセージであるsend_master2メッセージ数が3%程度削減されることがわかる。PIM/cのような階層型マルチクラスタでクラスタ間処理をクラスタ・コントローラで一括制御するような構成の並列マシンに対して、プロセスの割当て方法が適合した結果と言える。

5.2.2　動的負荷分散

動的負荷分散支援ハードウエアを使用して配線数136ネット規模の配線問題を8クラスタ（64プロセッサ）で実行した結果を表2に示す。実行時間が32秒で、ハードウエア機構を使用せずにソフ

表1．ディストリビュータメッセージ数と内訳
(136NETデータ使用。8クラスタで実行。
メッセージ数は平均値。)

		負荷均等化マッピング	通信局所化マッピング
send_master2メッセージ数		20791	20213
内訳	route	63	59
	estimate	2098	2080
	set_iep	18590	18050
	draw_line	118	115

表2．動的自動負荷分散方式比較
(クラスタ数：8CL;　配線数：136)

	プログラムコーディング量	実行時間
ソフトウェア制御方式	16行／1ゴール	58秒
ハードウェア制御方式	0行／1ゴール	32秒

トウェア負荷分散制御によって同条件で実行した58秒と比較して、約1.8倍に性能が向上した。これは、クラスタ負荷値をレジスタ参照するため、クラスタ負荷値のメモリ参照に比べ高速化できることによる。

また、ハードウエア支援動的負荷分散制御では、負荷分散先を判断・指定するためのソフトウェア制御文（1回の負荷分散処理に付き16行）を、動的負荷分散を指定するための述語1語に置き換えることができ、複雑なプログラミングが不要である。

クラスタ間の負荷が不均一な問題に対して、簡易なプログラミングで、より高性能な動的負荷分散が実現できることを実証した。

6．まとめ

プロセッサ8台で構成するクラスタを32クラスタ接続した、合計256プロセッサ構成の階層型並列推論マシンPIM/cを開発した。PIM/cには、動的な負荷分散処理支援のために各クラスタの負荷値を高速に通信する機構を設けた。我々は、PIM/c上に、ICOTの開発した並列化版LSI配線プログラムを実装して、クラスタ構成、および、動的負荷分散支援ハードウエアの効果を評価し、以下の結論を得た。

(1) 階層型並列計算機指向のプロセス割当て方法を導いた。

LSI配線プログラム移植に際して、クラスタ内プロセッサ間通信時間とクラスタ間通信時間のレーテンシ差を考慮し、クラスタ内の局所性を生かしたプロセスのマッピング方法に変更した。本評価では、クラスタに割り当てる格子の単位を大きくすることによって、クラスタ内の局所性を保つことを可能とした。その結果、960ネットの配線問題において、プロセス割当方法変更前に比べて、1.2倍に高速化した。さらに、256PE構成の実行で94.8倍の台数効果をあげることができた。

(2) 動的負荷分散ハード支援機構の効果をアプリケーションプログラムで実証した。

LSI配線プログラムにおいて、動的に負荷が変化する部分に、PIM/c独自機構である動的負荷分散ハード支援機構を用いて動的負荷分散を行った。その結果、ソフトウエア制御による動的負荷割当処理に比べて、1.8倍に高速化した。また、ハード支援機構を用いた動的負荷分散方式は、負荷分散のための複雑なプログラミングが不要である。以上により、応用プログラムにおける動的負荷分散ハード支援機構の有効性を実証した。

なお、本研究はICOTの再委託研究として行なわれた。

謝辞

本研究を行なうにあたり、御助言頂いたICOT内田俊一研究所長、同近山隆第1研究部長、神戸大学工学部瀧和男助教授(元ICOT研究室長)、NTT基礎研究所平田圭二氏(元ICOT研究室長)、並びに、プログラムを提供して頂いた（株）日立製作所日立研究所(元ICOT研究員)伊達博氏、ICOT関係各位に感謝致します。

参考文献

[1] （財）新世代コンピュータ技術開発機構：第五世代コンピュータ国際会議（第五世代コンピュータの研究開発成果）、1992.6

[2] 後藤厚宏、瀧和男、中川貴之、杉江衛：「並列推論マシンPIM/c」、情報処理学会第40回全国大会講演論文集(III), pp.1177-1178

[3] （財）新世代コンピュータ技術開発機構：「無償公開プログラムリスト」,1992.6

[4] 伊達博、大嶽能久、瀧和男：「並列オブジェクトモデルに基づくLSI配線プログラム」、情報処理学会論文誌、Vol.33,No.3,pp.378-385(Mar.1992)

[5] 井門徳安、前田浩光、垂井俊明、中川貴之、杉江衛：「並列推論マシンPIM/c－負荷分散支援機構－」情報処理学会第40回全国大会講演論文集(III), pp.1181-1182

[6] M.Sugie,M.Yoneyama,N.Ido and T.Tarui："LOAD-DISPATCHING STRATEGY ON PARALLEL INFERENCE MACHINES", Proc. of FGCS'88 Vol.3, pp.987-993

[7] （財）新世代コンピュータ技術開発機構：「並列LSI配線プログラム」、第五世代コンピュータ国際会議1992デモンストレーション説明資料、pp.47-50