TR-671

並列推論マシンPIM/cにおける自動負荷分散支援機構

中川　貴之、杉江　衛（日立）

July, 1991

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03)3456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

並列推論マシンＰＩＭ／ｃにおける自動負荷分散支援機構

Automatic Load Ballancing Features on PIM/c (Parallel Inference Machine / model c)

中川貴之　　杉江衛　（（株）日立製作所　中央研究所）
Takayuki NAKAGAWA　　Mamoru SUGIE
(Central Research Laboratory, Hitachi Ltd.)

[1] 概要

計算機の並列処理の利用技術の可能性を探るために、並列推論マシンＰＩＭ／ｃ (Parallel Inference Machine/model c)[1]の研究では自動負荷分散技術の開発に力を入れている。ＰＩＭでは、並列計算機アーキテクチャにおいて、２つの流れとなっている、メモリ共有（密結合TCMP）とメモリ非共有（疎結合LCMP）の両結合方式における得失を考慮して、密結合クラスタを疎結合する階層結合方式を採用している。ＰＩＭ／ｃでは、それぞれの結合階層で、高速な自動負荷分散支援ハードウェアを実現した。

本稿では自動負荷分散技術の中でも、特に現状の並列処理研究の動向から特定のプログラムに依存しない動的負荷分散技術の評価手法が不可欠と判断し、負荷分散パラメタと細粒度並列計算モデルを提案する。

[2] 密結合と疎結合の得失

並列マシンの構成法にはMulti-PSI[2]のような疎結合と、大型汎用機やスーパーコンピュータのような密結合の２つがある。

疎結合では台数拡張性があり、1000台規模の計算機システムが構成できるが、メッセージ作成やコピー回数を最小限にする共有データ管理の手間により、レスポンスに100サイクルオーダーの時間を要する。

密結合ではキャッシュの導入により、レスポンスタイムを１サイクルオーダーに短縮でき

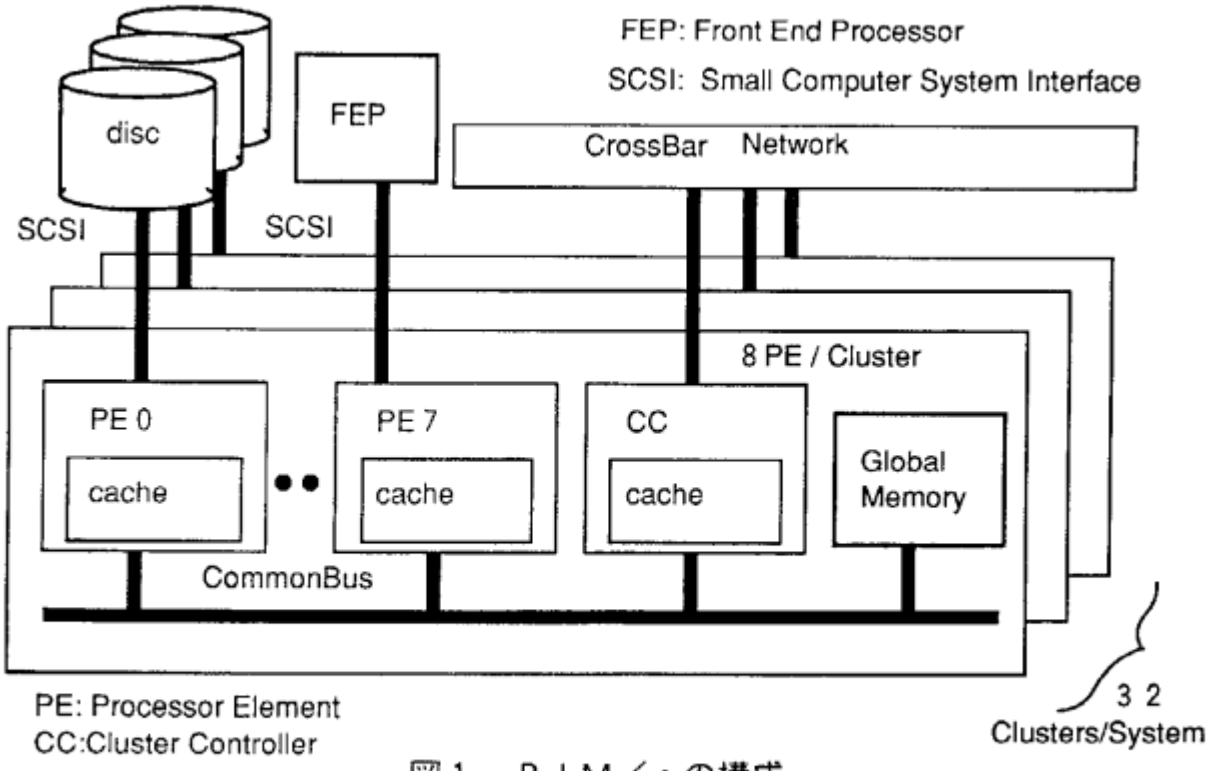


図１　ＰＩＭ／ｃの構成

るが、キャッシュを構築しやすい共有バス環境では、１０台程度で共有バスのスループットがボトルネックになる[3]。そこで、ＰＩＭ／ｃでは図１のように、密結合クラスタを疎結合した階層結合方式を採用している。表１に両者の得失をまとめる。

表１　結合方式の利害得失

結合方式	レスポンス	台数拡張性
密結合	○	X
疎結合	X	○

［３］　全自動負荷分散の可能性

自動負荷分散にはコンパイラによる静的な負荷分散と、実行時に行う動的負荷分散があるが、スーパーコンピュータの普及理由を振り返ると、以下の要因を挙げられる。

１）数値計算分野の標準言語FORTRANの採用
２）高速な記憶手段の提供
a) ベクトルレジスタによる多項演算の高速化
b) SRAM主記憶の採用
c) 拡張記憶による入出力の高速化
３）自動ベクトル化
処理時間の大部分を占める繰り返し演算の高速化アルゴリズムの存在により、まったくプログラムを書き換えずに高速化

一方で、並列マシンの現状には以下の問題がある。

１）並列処理の標準言語がない
２）高速化手段はマシン依存である
３）高速化手法にはプログラマの指示が必要

標準言語が存在しない現状では、まずＫＬ１(Kernel Language 1)[4]のような並列処理に適したプログラミング言語を普及させ、さらに、キャッシュやディスクへの並列アクセスのような高速化手段を言語にチューニングしていくことが重要だが、当分の間、高速化手法は半自動であると考えられる。なぜなら、並列化による高速化手法は、負荷分散オーバヘッドの増加による低速化の要因をかかえているからである。

自動負荷分散にはコンパイラが行なう静的負荷分散と、ハードウェアとシステムライブラリが実行時に行なう動的負荷分散があるが、静的負荷分散には以下の長所がある。

a) プログラムの大域的な性質を捉えて最適化することができる
b) コンパイラが実行できる固定的な処理については実行時オーバヘッドを削減できる

半面、以下の短所も合わせ持つ。

c) 台数構成が変わるとリコンパイルが必要
d) 入出力等で実行時に決まるデータに依存した負荷の大きさに関してはプログラマの指示が必要
e) 負荷の実行順序は、プログラマにもわからない

ＫＬ１言語はユーザを同期処理や負荷の実行順序を指定する手間から解放するが、同期処理時間に依存する、負荷の実行順序は、高速化のために再現性も保証しないので、実行してみないとわからない。

従って、ＫＬ１プログラムの処理系が行なうような細粒度の処理では特に、動的負荷分散が必要であると考える。

ＰＩＭ／ｃでは、動的負荷分散のオーバヘッドを削減するハードウェアを備えているので、６４台程度までの全自動動的負荷分散を試みようと準備中である。これには、以下の理由がある。

１）プログラムに依存した評価も重要であるが、並列処理ハードウェアとしての生の特性を見たい。
２）オーバヘッドを十分小さくするハードウェア支援と、無駄の少ないアルゴリズムが作れれば、プログラム自体のもつデータアクセスの局所性によって、データ転送オーバヘッドが飽和するポイントが出ないかを見たい。
３）密結合と疎結合のトレードオフポイントを見たい

この試みは、プログラムの特性を無視するところから出発するが、逆に、細粒度のまま、高速化の指標をパラメタ化して、スーパーコンピュータのベクトル長に相当するものを見い出し、これに適したプログラミングモデルを打ち立てることを目的とする。

［4］　ＰＩＭ／ｃの動的負荷分散支援

以下に、ＰＩＭ／ｃの結合階層毎の動的負荷分散アルゴリズムとその支援ハードウェアを説明する。その前にＰＩＭ／ｃにおける負荷分散の考え方を以下にまとめておく。

1）集中を排除する
動的負荷分散の一番簡単な方法は、負荷を一箇所のテーブルに管理し、そこに１個ずつ出し入れしていくことである。しかし、このような集中は膨大なテーブルの出口と入り口のみを活性化し、負荷分散処理自体の均等分配を配慮していない。

2）無駄な処理を排除する
並列処理ハードウェアであっても、無駄な処理ばかりをしていては、全体の挙動を評価する際の視点を曖昧にするだけである。

4.1 疎結合における動的負荷分散支援

4.1.1 疎結合の負荷分散アルゴリズム

疎結合のプロセッサ間処理ではレスポンスが遅いので、負荷分散アルゴリズムとしては負荷が全く無くなってからではアイドルタイムが長くなる。また、負荷ばらつきが大きく、分散が必要な時には負荷の少ないプロセッサが多いと考えられ、マルチプサイの報告[2]にもあるように、負荷を要求する方式では負荷要求が集中し、要求した負荷が得られない場合が多いと考えられる。以上の２点から、負荷の多いプロセッサから分配のきっかけを作る、センダ･イニシエート(Sender-Initiate)方式を提案する。

センダ･イニシエート方式は無駄な負荷分配が発生しうるので、これを削減するために負荷分布の知識を前提とするが、図２に示す負荷分散方式の例のように、大域的な分布情報は収集の手間が大きいうえ、状況の変化と共に不正確になる。そこで、図３のように乱数で決定したレシーバ候補と自分の負荷量を比較して無駄な分配を削減するスマートランダム(Smart-Random)方式[5]を採用する。

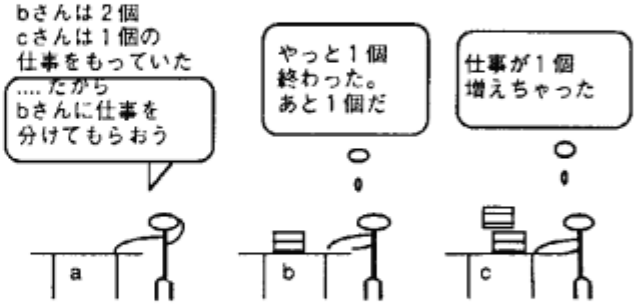


図 2　避けたい負荷分散方式

閾値付きスマートランダム方式の拡張アルゴリズムは以下のとおりである。

・定期的に、閾値以上の負荷を持つ忙しいクラスタから分配処理を試みる
・乱数で決定した相手クラスタが自分より多忙なら分配しないで処理を終了
・自分よりある一定値以下の負荷しか持たない程度に暇なら負荷を１個だけ送り付ける

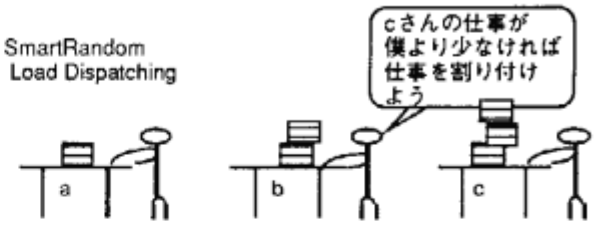


図 3　スマートランダム方式

オリジナルのスマートランダム方式では、分配時の判定基準が、単なる大小関係によっていたのを、この方式は一定値の差分を要求している点で、余分な分配を削減できるように拡張してある。１度に分配する負荷を１個では無く複数個にすると無駄な分配にもなりうるので、現時点では細粒度の負荷分散を目指している。また、絶対値でなく相対値で判定するのは、ある程度暇なクラスタが発生するのを許容する効果が期待できる。

この方式はシステム全体における負荷分布の知識を必要としない。また放送通信のような、台数の少ないシステムに有利な機能を必要としないのが利点である。

4.1.2 疎結合の支援ハードウェア

相手クラスタの負荷を高速に知るために、ＰＩＭ／ｃでは筐体内に限って、図４のように負荷問い合わせメッセージをバイパスする、以下のハードウェアを設けた[6]。

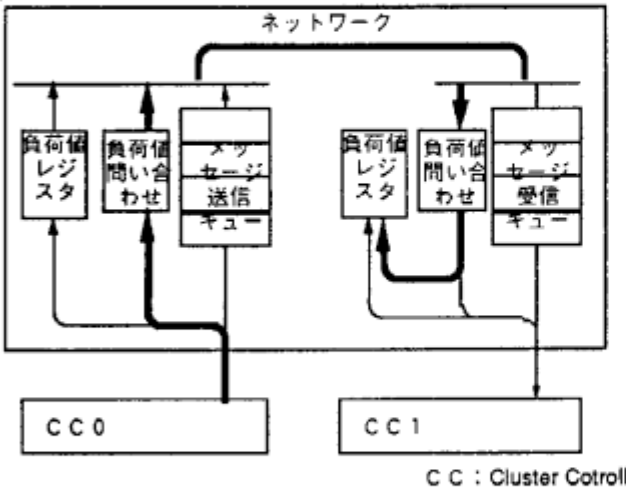
1）ネットワーク中に8ビットの負荷値レジスタＣＬＲ(Cluster Load Register)
2）負荷値問い合わせメッセージのバイパス

上記ハードウェアにより、相手プロセッサに割り込むことなく、正確な負荷量を迅速に知ることが出来る。このための追加処理として、各クラスタがクラスタ内の負荷量をCLRに自律的かつ定期的にセットするものとする。

表２に負荷値レジスタの効果をまとめる。

表２　負荷値レジスタによる高速化

方式	レスポンス
通常メッセージ	プロセッサやキューの忙しさに依存して遅い
負荷値問い合わせ	１１サイクル



ＣＣ：Cluster Cotroller

図４　疎結合動的負荷分散支援ハードウェア

4.2 密結合における動的負荷分散支援

4.2.1 密結合の負荷分散アルゴリズム

密結合ではレスポンスが良いので、無駄な負荷分散が発生しないように、負荷が全く無くなってから負荷分散を要求するレシーバ･イニシエート(Receiver-Initiate)方式を採用する。密結合であっても負荷分布の大域的な情報を更新する手間を避けるために、図５に例を示すような匿名通信(Annonymous/1 to Any - Communication)方式を用いる。匿名通信方式の概要は以下のとおりである。

･暇なプロセッサから負荷要求処理を行なう
･最初に気づいた任意のプロセッサが負荷分配処理を行なう

図５　匿名通信方式

この方式もシステム全体における負荷分布の知識を必要としない。またレスポンスタイムが短いために、無駄な負荷分散を避けて、負荷分散が必要になってから通信しても十分であると考えられる。

4.2.2 密結合の支援ハードウェア

ＰＩＭ／ｃでは、図６のように、全てのプロセッサに高速に要求を伝える、以下のハードウェアを設けた。

1）　放送書き込み機能のついた要求フラグの専用レジスタ
2）　要求フラグのall-0チェックと分岐処理を１サイクル(50nsec)で実行する命令

このハードウェアを以下の手順で使うことで、任意のプロセッサによる迅速な応答が期待できる匿名通信を実現できる。

a) 負荷の全く無くなったプロセッサが、全プ

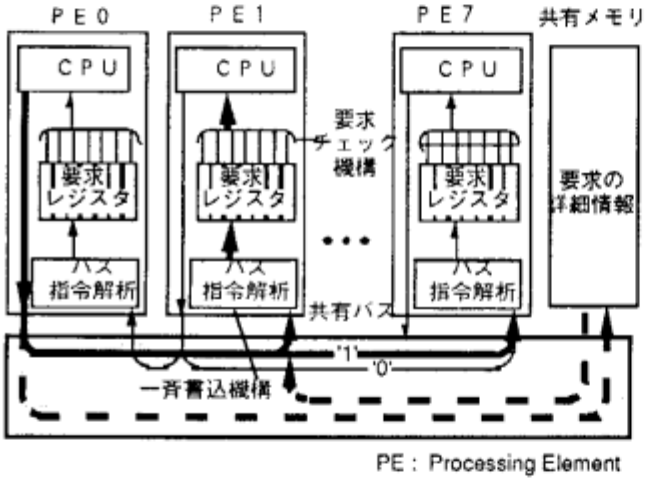
ロセッサに要求を発行する
b) 最初に気づいた任意のプロセッサが、全プロセッサに対する要求のリセットを行なう

この時、共有メモリ上の詳細情報として要求プロセッサのビットマップをFetch&Or命令で更新することにより、排他制御を実現できる

以下に支援ハードウェアの高速化効果をまとめる。

表3　匿名通信ハードウェアによる高速化

方式	レスポンス	共有バスアクセス数
キャッシュ	9サイクル	8回
要求レジスタ	1サイクル	2回



PE : Processing Element

図6　密結合動的負荷分散支援ハードウェア

4.2.3 支援ハードウェア方式の比較

上のような匿名通信を、通常のスヌープキャッシュ(Snooping Cache) を用いた固定アドレスのポーリングによってサポートすると、1回の要求イベントの書き込み毎に全プロセッサにキャッシュミスが発生し、本提案による場合の2回に比較して、共有バスの使用回数を著しく増加させ、システム性能を下げることは明らかである。

放送書き込み機能をキャッシュに持たせる方式もある[8]。しかし、汎用的であるためにハードウェアコストの割に十分な高速化が得られない。また、一般のメモリアクセスに対して、共有バスの使用時間が増加する傾向があり、アクセスローカリティを期待する機構にはそぐわないとの評価結果から専用レジスタを選択した。

同じく専用レジスタを設ける方式として、ＣＲＡＹ－Ｘ／ＭＰのコモンレジスタがある。これと比較すると、ＰＩＭ／ｃでは、大域的な読みだし機構を必要とせず、レジスタ幅も小さいので、ハードウェアが軽量である。また、共有バスを使うことによって専用線も不要である。

また、汎用計算機に用いられている、外部割り込みインタフェースを導入する方式もある。しかし、いつでも他のプロセッサから割り込めるようにするには、undo機能が必要である。undo機能は、通常の書き込み処理において、書き込み前のデータを保持するためのオーバヘッドが大きい。推論マシンでは、このような他律的割り込みの概念を排除して、スリットチェックという自律的な割り出しポイントを設けているので、このスリットチェックポイントでの割り出しとしてサポートするのが妥当である。

通常の割り込みハードウェアでスリットチェックをサポートすると、割り込みマスクを毎回on/offする手間が繁雑である上に、無駄である。

4.3 二階層動的負荷分散の結合方式

以上の2階層の動的負荷分散方式を、ＰＩＭ／ｃでは以下のように結合する。

1）各プロセッサは一定間隔で、自律的に自プロセッサの実行待ちゴール数を固定領域に書き込む。ネットワークに接続を有するクラスタコントローラが一定間隔で、これらの総和をとることにより、プロセッサに割り込むことなくクラスタ内負荷量を求めて、ＣＬＲに書き込む。

2）各プロセッサ負荷量の記録領域はプロセッサ側の書き込み時のキャッシュミスを削減するために、排他的なキャッシュブロックとする。クラスタコントローラは読みだすだけであるので、プロセッサは無効化のみのコストで書き込みができる。

3）負荷分散が不要と判断された場合の無駄を避けるために、負荷量を比較して、必要と判断されてから、クラスタコントローラは匿名通信により分配する負荷をクラスタ内プロセッサに要求する。

［5］　計算モデル

5.1 負荷分散パラメタ

得られる負荷分散方式はなるべく少ないパラメタにより制御できなければ、結局使いにくいものになる。現時点での負荷分散パラメタは以下のとおりである。

・プロセッサ負荷量、クラスタ負荷量ＣＬＲを設定し、検査する時間間隔:Td
・負荷分配を試みる閾値:CLR0
・負荷分配を決定する差分:CLRd

CLR0の値にはＰＩM-Rで最適であった、値５にプロセッサ台数８を加えた、１３程度を目安とする。値Tdを変えることにより、クラスタ間のゴール投げだし比率を制御できる見込みである。差分CRLdを小さくすることは、負荷分散の機会を増やし、Tdを小さくすることに相当すると仮定すると、CLRd*Tdを一定（例えば１６）にして比較する事ができる。

5.2 計算モデルのパラメタ

一方、プログラムのひな形となる計算モデルとしては、特定問題に依存しないために、以下のパラメタを用意する。

1）負荷の総量
L=SUM(Li)
2）プロセッサ間の負荷量のばらつき
V=SUM((Li-<Li>)*(Li-<Li>))
3）負荷粒度のばらつき

図７に例を示すように、分配されるゴールは１個であっても分岐のないノードは分割できないので、分散される負荷量は不定である。分配元も２個以上のゴールを持てば任意である。

負荷xから負荷yがとれる確率は一様に1/(x+1)と仮定する。

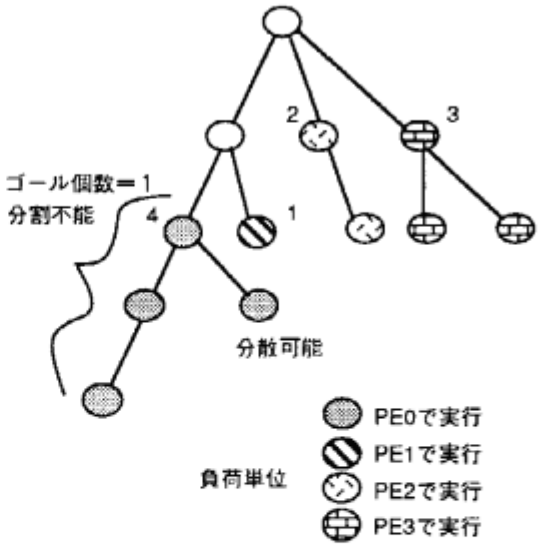


図７　負荷分散の対象問題例

5.3 負荷分散の効果の評価方式

次の比較により行う。

1）初期分配のみの時の処理時間
（最初のＮ個はコンパイラで割り付けを仮定）
T_init＝MAX（Ｌi）{i=1,2,..N}
2）動的負荷分散時の処理時間
T_dyn＝（Ｌ＋Ｍ＊ｐ）／Ｎ
負荷分散コストｐ
負荷分散回数　Ｍ

このような負荷状態の遷移過程は各プロセッサにおける負荷量{Li}の遷移として表される。この時間経過の一例を図８に示す。

図８の”=>”は分散処理を表し、
”->”は時間経過を表すものとする。

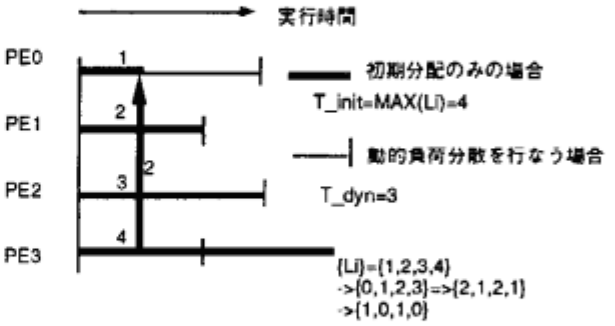


図８　負荷分散実行例

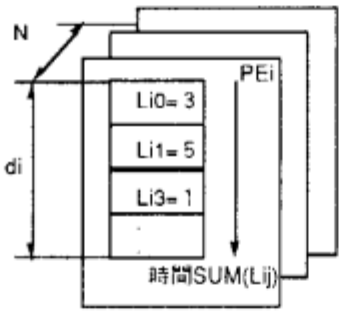
5.4 計算モデルの実行

以上のパラメタで、以下のような処理を行わ

せる。

1）図９のような一様乱数列Lijを生成する
2）同じく一様乱数diによりプロセッサiの生成する負荷量を決定する
3）各プロセッサは負荷量に見立てたLij＊単位時間だけカウンタをデクリメントする
4）負荷要求が発生したら、単位時間の切れ目で、次に使う予定であったLijの値を負荷量に見立てて分配する
7）それぞれのプロセッサは一定時間毎にデータアクセスを、外乱として与える。データアクセスは局所性、書込比率、共有率、頻度といったパラメタを持つ。

データ共有率および台数Nに比例したメッセージ転送あるいはキャッシュ間ブロック転送が発生する。



Lij: プロセッサiがj番目に生成した負荷粒度
di: プロセッサiが生成する負荷の数

図９　粒度のバラツキを加味した計算モデル

［６］　おわりに

本稿では動的負荷分散を、現状の並列処理研究の動向から不可欠と判断し、細粒度動的負荷分散への適用を目的とした、負荷分散パラメタを用いた計算モデルを提案した。

また、２階層の結合方式にそれぞれの特長を加味した、負荷分散方式とその支援ハードウェアの紹介を行なった。

ＰＩＭ／ｃのハードウェアは、現在１６プロセッサの部分試作機が稼働中で、ＫＬ１処理系の実装作業を実施中である。今年度に製造する２５６台のシステムでは、密結合８プロセッサは匿名通信ハードウェア、疎結合８クラスタはスマートランダム方式支援ハードウェアを有する。

今後の課題として、上記の計算モデルを使って、実装した支援ハードウェアの効果を評価していく予定である。

また、以上の支援ハードウェアを持たない４筐体間の負荷分散はプログラムの知識を前提としたコンパイラによる負荷分散を準備中である。その場合、データの転送量と転送回数の最適化が主な目標と考える。

［７］　謝辞

本評価モデルの構成にあたり、助言をいただいた同僚である垂井、井門の両氏に感謝したい。なお、本研究はＩＣＯＴからの委託研究の一環として、実施された。

［８］　参考文献

1) Goto et al; "Overview of the Parallel Inference Machine Architecture(PIM)";Proc of the FGCS vol1 1988
2)古市ほか、" スタック分割動的負荷分散方式とマルチＰＳＩ上での評価" ,Proc.of KL1 Programming Workshop '91
3) 松本ほか、" KL1のメモリ参照特性に適した並列キャッシュ機構" ;データフローワークショップ;1987-10
4) Ueda et al;"Guarded Horn Clauses",TR209,ICOT,1985
5) 杉江ほか、"Load Dispatching Strategy on Parallel Inference Machines",Proc. of FGCS Vol.3 1988
6) 井門ほか、" 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、２Ｌ—４
7) 中川ほか、" プロセッサ間ソフトウェア割り込み処理を高速化するスリットチェック機構" 計算機アーキテクチャ研究会、1989-7
8) McCreight,E. ;"The Dragon Computer System An Early Overview";Xerox Corp.,Sept.,1984.