1R-5　ＰＩＥのパイプライン・ゴール書き換え方式とゴール・マルチキャスト方式

小池　汎平　，　田中　英彦　，　元岡　達

（　東京大学　　工学部　）

1．はじめに

高並列推論エンジンＰＩＥ［１］はゴール書き換えモデルに基き、論理型言語をＯＲ並列に処理する。ゴール書き換えモデルでは、ＯＲ並列処理によって生じる多重環境を各プロセッサがローカル・メモリに独立に持つため、高並列処理に向くという利点がある反面、１回の推論処理ごとに必要となるコピー処理のオーバヘッドが大きくなるという欠点がある。我々は、これまでに、コピーのオーバヘッドの実行時間への影響を、パイプライン処理によってなくすパイプライン・ゴール書き換え方式［２］及び、ゴール・マルチキャスト方式［３］を提案してきた。パイプライン・ゴール書き換え方式は、ゴールが１回書き換えられるたびの遅延を、単一化時間と縮退時間の和から単一化時間のみに減少させる。また、ゴール・マルチキャスト方式は異る定義節を用いた書き換えごとの遅延をほぼ０にする。本報告では、これらの方式のシミュレーション評価結果を示し、更に、パイプライン・ゴール書き換え方式と逐次型推論マシンとの逐次的な問題における速度比較を行なう。

2．シミュレーションによる評価

パイプライン・ゴール書き換え方式を用いる／用いない、ゴール・マルチキャスト方式を用いる／用いない、の４通りの組み合せについて、シミュレーションを行なった。単一化プロセッサ［４］のタイミングは試作単一化プロセッサに基き決定し、ネットワークは接続要求があると遅延時間なしでプロセッサ間の接続を行なうものとした。このようなネットワークは、動的負荷分散網［５］により実現できる。ベンチマーク・プログラムとしてパタン・マッチングを用いた８クイン・パズル（８qa）、リスト表現を用いた６クイン・パズル（6q）、リストの反転（nrev30）、クイック・ソート（qsort50 ）を用い、実行時間、最初の解を得るまでの時間、全プロセッサの稼動時間の総和、平均並列度、単一化の平均時間、縮退の平均時間を求めた。結果を表１に示す。表より、以下のことがわかる。

・nrev30の実行速度はパイプライン・ゴール書き換え方式を用いることにより、10倍になる。
・８qaの実行速度はパイプラインとマルチキャストを併用

表１　シミュレーション結果

programs		methods			
		non.pipe. non.mult.	pipe. non.mult.	non.pipe. mult.	pipe.mult.
8qa	exec. clk.	30648	23309	N.A.	13856
	1'st ans.	17682	9730	N.A.	4944
	total clk.	3243152	3197278	N.A.	3091467
	total wait	0	78358	N.A.	119364
	av. par.	105.8	137.2	N.A.	223.1
	unif. clk.	383.5	406.6	N.A.	407.5
	red. clk.	641.2	580.2	N.A.	512.4
6q	exec. clk.	17551	8437	15328	6059
	1'st ans.	15120	6797	14157	5719
	total clk.	711665	748588	734951	785233
	total wait	0	93821	0	105803
	av. par.	40.7	88.7	48.1	129.6
	unif. clk.	44.1	56.9	46.0	59.2
	red. clk.	194.5	171.5	193.7	169.5
nrev30	exec. clk.	346794	33895	351601	38563
	total clk.	351800	361603	356607	365978
	total wait	0	9803	0	9371
	av. par.	1.01	10.7	1.01	9.5
	unif. clk.	47.9	66.2	49.9	67.6
	red. clk.	647.6	648.2	647.3	647.6
qsort50	exec. clk.	437644	57436	426775	59875
	total clk.	545003	403124	546562	390757
	total wait	0	46718	0	31100
	av. par.	1.25	7.02	1.28	6.53
	unif. clk.	51.3	86.7	52.3	68.0
	red. clk.	787.6	525.3	786.9	498.7

することで 2.2倍になるが、最初の解を得る速度は 3.5倍に向上した。
・縮退の平均時間は８qaとqsort50 で20～36％減少する。これは不要な縮退処理が除去されるためである。

３．逐次型推論マシンとの実行時間の比較

ここでは、パイプライン・ゴール書き換え方式を採用した場合のＰＩＥ（以下、単にＰＩＥと呼ぶ。）と、逐次型推論マシンの中で現在最も高速といわれているWarren の命令セット［６］を用いたマシン（以下、ＷＩＭと呼ぶ。）との、逐次的な問題における速度の比較を試みる。

ＰＩＥにおける単一化・縮退の実行は、ＷＩＭでは get/put/unify命令の実行に相当する。より正確には、ＰＩＥにおける単一化はＷＩＭにおける get命令とこれに続いた unify命令に対応し、ＰＩＥにおける縮退のうち、定義節の本体部の縮退はＷＩＭにおける put命令とこれに続いた unify命令に対応する。これ以外の縮退処理、つまり、前ゴールから引き継がれたリテラル・構造データの縮退処理は、ＷＩＭには対応する処理のない、ゴール書き換えモデル特有のオーバヘッドである。これらの関係を図１に示す。

さて、パイプライン・ゴール書き換え方式を用いることにより、１回の推論（ゴールの書き換え）当りの遅延時間は単一化の処理時間のみとなる。これに対し、ＷＩＭでは、１回の推論（命令列の実行）当りの処理時間は全命令実行時間の合計になる。したがって、１回の推論当りの時間は、ＰＩＥのほうが、 put命令・ unify命令に相当する処理の時間がパイプラインによって隠される分だけ短くなる。図１で、影付けされた部分は実行時間に影響を与えない処理である。

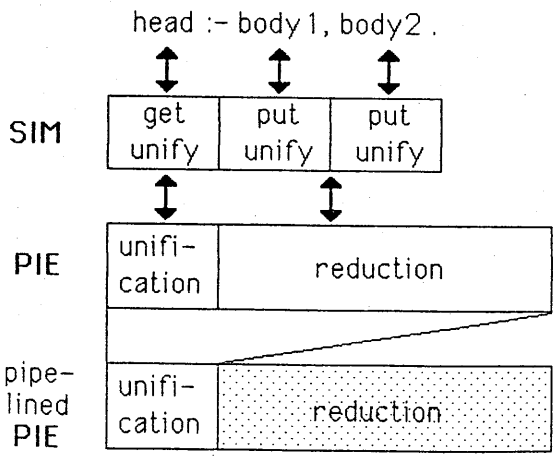

図１　処理の対応関係と実行時間の比較

次に、ゴール側が変数、定義節側が構造データであったときの単一化の手間を比較する。ＰＩＥでは単にポインタを変数に代入するだけで定数オーダであるのに対し、ＷＩＭでは、 get命令に続く unify命令を実行し、構造データのコピーを行なうため、構造データの大きさに比例した手間が必要である。ＰＩＥでも、構造データのコピーを行なうが、コピーは縮退処理の時まで遅延される。したがって、構造データのコピーはパイプラインに隠され実行速度に影響を与えない。更に、単一化が途中で失敗したときは、遅延された構造データのコピーは実行されず、無駄な処理を省くことができる。

ＷＩＭにおける単一化は、ＰＩＥにおける単一化に比べ、メモリ管理などの制約から、
・変数同志の単一化の際にアドレスの大小比較が必要
・変数の束縛時に行なうトレイル処理するかどうかの判断
が比較的複雑
など、複雑化している。したがって、個々の引数当りの単一化の手間はＰＩＥのほうが小さくできる可能性がある。

以上の議論より、パイプライン・ゴール書き換え方式を用いることにより、ＰＩＥは、逐次的な問題においても、ＷＩＭと同等、あるいは、それ以上の速度を出し得ると結論づけられる。しかしながら、より詳細な評価は、実際にハードウェアを試作し、ある程度大きな応用プログラムを実行してみて行なうべきであることはいうまでもない。

一方、ＷＩＭでは処理のうち不要なものがコンパイル時の最適化ににより取り除かれるが、ＰＩＥでの処理は定義節の内部形式がデータと同一形式をしており、プログラムと１対１に対応しているという点で、インタプリティブである。最適化コンパイルによる不要な処理の除去はこれからの検討課題である。

４．おわりに

本報告では、パイプライン・ゴール書き換え方式とゴール・マルチキャスト方式により、ＰＩＥのゴール書き換え処理が高速に実行されることを示した。また、パイプライン・ゴール書き換え方式により、ＰＩＥが逐次的な問題においても、逐次型推論マシンと同等以上の速度を出し得ることを示した。

＜参考文献＞

［１］Moto-oka et al “The Architecture of A Parallel Inference Engine -PIE- ”, Proc. of the International Conference on FGCS'84, pp 479-488, Sept.1984.
［２］小池　他，“ＰＩＥの単一化プロセッサのパイプライン接続について”，第30回情処全大，2c-8，1985.
［３］小池　他，“ＰＩＥにおけるゴール分配方式”，第31回情処全大，2c-8，1985.
［４］Yuhara et al “A Unify Processor Pilot Machine for PIE ”,Proc. of the Logic Programming Conference '84,7-2,Tokyo,March 1984.
［５］坂井　他，“動的負荷分散を行う相互結合網”，信学技報，1985年 8月.
［６］D.H.D.Warren, “An Abstruct Prolog Instruction Set”, SRI TN309,1983.