7T-4

トーラス型マルチプロセッサシステム上でのProlog並列処理手法

真栄田 保　　川口 剛　　喜屋武 盛基
(琉球大学工学部)

1．まえがき

Prologプログラムを並列計算機上で実行させることにより、実行の高速化をはかろうとする試みが多くなされてきている[1],[2]。本稿では、高並列度を実現するのに適した構造をもつトーラス型マルチプロセッサシステム上でのPrologの並列処理手法を提案する。Prologの並列処理方式には、OR並列、AND並列、引数間並列などの方式があるが、実現の観点からOR並列処理方式が最も有望視されている[1],[2]。本稿で提案する手法もOR並列処理方式を用いる。

2．トーラス型マルチプロッセサシステム

図1に示すように、本研究で用いるトーラス型マルチプロセッサシステムは、m^2個のPE、m個のCU（コントローラ）及びHOST（ホスト計算機）から構成される。PEは隣接する4個のPEと結合されている。また、各行のm個のPEに対して1個のCUが割り当てられ、CUは自分の配下にあるm個のPEと結合されている。更に、すべてのCUはHOSTと結合されている。

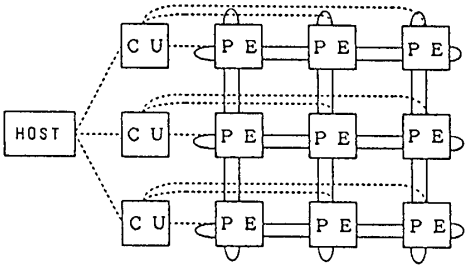


図1．トーラス型マルチプロセッサシステム

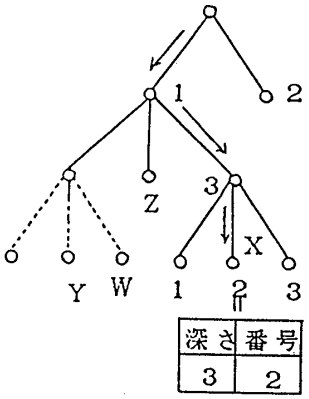
3．並列処理手法

3.1　PE間でのノードの受け渡しの方法

本手法では、図1に示したm^2個のPEが並列に木の探索を実行する。負荷分散のため、隣接するPE間でノードの受け渡しを行うことが必要になるが、本手法ではノードのすべての履歴を通信するのではなく、探索木におけるノードの深さと左から数えて何番目かという情報のみを通信する。この情報をノード情報と呼ぶ（図2）。

また各PEは、探索木上で自分がたどってきたパス（道）を隣接するPEに知らせるため、単一化するノードを選ぶたびにこのノード情報を隣接するPEに伝える。そしてこれを受け取ったPEは図3のような表にノード情報を記憶していく。PEはこの表を参照することにより、隣接するPEがどのノードを処理中かを知ることができる。例えば図3の表から、右隣りのPEがノードYを処理中であることがわかる。

更にPE間でのノードの受け渡しは次のように行われる。図2に矢印で示されたパスをたどってきたPE（以後このPEをPE(i)と呼ぶ）が、ノードXで単一化に



深さ	番号
3	2

図2．ノード情報

隣接するPE

深さ	左	右	上	下
1		1		
2		1		
3		2		

図3．隣接PEのパス情報

失敗し右隣りのPEにノードを要求したと仮定する。このとき、右隣りのPEがキューの中にノード（実際にはノード情報）をもつ場合でも2つの状況が考えられる。

（i）ノード情報を受けたPE(i)がバックトラックの後、ただちにこのノードを処理できる場合（例えば右隣りのPEがキューの先頭にノードZをもつ場合）：この場合、右隣りのPEはノード情報(2,2)を送る。PE(i)は、自分がたどってきたパスの情報と右隣りのPEのパス情報を比較し、両者が一致する深さ1までバックトラックした後、送られてきたノード情報(2,2)に従ってノードZを選ぶ。（PE(i)は矢印のパスをたどってきたのだから、実線で描かれた部分木をもっている。）

（ii）ノード情報を受けたPE(i)がバックトラックの後、ただちにこのノードを処理できない場合（例えば右隣りのPEがキューの先頭にノードWをもつ場合）：この場合、右隣りのPEはPE(i)にノードを要求された時点ではWのノード情報を送らず、PE(i)を自分のslaveにして自分と同じパスをたどらせる。PE(i)は深さ1までバックトラックした後、右隣りのPEのパス情報に従ってノードを選んで行く。そして右隣りのPEは、PE(i)が次に深さ3のノードを選ぼうとする段階でキューの先頭のノードがWならば、これをPE(i)に送る。（しかし例えば、ノードYが終端され、右隣りのPE自身がWを処理することもありうる。）

上記の（i）、（ii）を区別することにより、無駄なノードの受け渡しを避けることができる。

3.2　PEの動作

PEは、次に処理すべきノードをどのような方法で選ぶかによってcontroled、individual、master、slaveの4つのタイプに分類される。controledはHOSTからの指示に従ってノードを選ぶ。slaveはmasterから送られてくる情報をもとに、masterの探索を追従する。masterとindividualは、自分がもつ部分探索木をもとに深さ優先探索に従ってノードを選ぶ。masterとindividualの違いは、前者が自分の指示に従うslaveをもつという点である。最初すべてのPEはcontroledである。そして各PEはある時点でindividualに変化し、再びcontroledに戻ることはない。以下にPEの動作を4つのタイ

Parallel Execution Scheme of Prolog Programs on a Torus Machine.
Tamotsu MAEDA, Tsuyoshi KAWAGUCHI, Seiki KYAN, University of the Ryukyus.

プごとに記述する。

（Ⅰ）controledの動作

最初すべてのＰＥはcontroledであり、すべてのＰＥが一斉に根ノード（与えられたゴール節）を処理する。そして、この結果生成されるノードがＨＯＳＴによってm行に分配される。もし１つの行に複数のノードが割り当てられるなら、これらのノードが左端のＰＥから順に１個ずつ割り当てられる。このとき単独にノードを割り当てられたＰＥはcontroledからindividualに変化し、以後の動作は（Ⅱ）に従う。そしてこの後ＨＯＳＴから割り当てられるノードはcontroledのみに分配される。

（Ⅱ）individual、masterの動作

次の（１）～（８）を繰り返す。

（１）ＨＯＳＴからの繰り返し指令を受けたとき、もし自分が次に単一化すべきノードをもたないなら、隣接するＰＥの中で「少なくとも１個のノードをもち、しかもcontroled、slaveでないＰＥ」にノードを要求する。

（２）隣接するＰＥからノードを要求されるならば以下を行う：自分が次に処理するノード以外にキューの中にノードをもち、しかもノード要求してきたＰＥがただちにキューの先頭のノードを処理できる場合には、このノード情報を送る；それ以外の場合には、ノードを要求してきたＰＥを自分のslaveにする。

（３）ノードを要求してきた先のＰＥからノード情報のみが送られてくれば、これを次に処理するノードとし（４）へ進む。一方slaveになるよう指示されてくれば、このＰＥのパス情報に従ってバックトラックしたときのノードを次に処理するノードとし（Ⅲ）の（４）に移る。

（４）ＨＯＳＴから実行指令があれば、現在自分がもつノードに対して単一化を実行する。

（５）（４）の結果、単一化に成功しノードv_1，…，v_kが生成されれば、次に単一化すべきノードをv_1とし残りのノードをキューの末尾に入れる。また、ノードが終端され（単一化に失敗するか解が見つかり）キューの中にノードがあれば、キューに最後に挿入されたノードを次に単一化すべきノードとする。

（６）自分がmasterであるときのみこのステップを行い、individualならば（７）へ飛ぶ。次に単一化すべきノードがないならば自分の指示を受けているすべてのslaveをindividualに変える（master-slave関係を解消する）。一方、次に単一化すべきノードがある場合には、２つの場合ごとに以下の処理を行う：キューの中にもしノードがあり、しかもslaveが次に処理しようとするノードの深さがキューの先頭のノード（深さが最も浅いノード）の深さに一致する場合には、キューの先頭のノードをslaveに送り、このslaveをindividualに変える；その他の場合にはslaveにmaster-slave関係の継続を伝える。

（７）次に単一化すべきノードがない場合には、ＨＯＳＴと隣接するＰＥにidle信号を送る。ノードがある場合には、このノード情報を隣接するＰＥに送るとともに、ＨＯＳＴにbusy信号を送る。（ＨＯＳＴは、すべてのＰＥからidle信号が送られてきた場合のみＰＥに停止指令を送り、その他の場合には繰り返し指令を送る。）

（８）隣接するＰＥからノード情報が送られてくれば、これを図３のような表に記録する。

（Ⅲ）slaveの動作

各ＰＥが同期をとりながら動作していることを明かにするため、（Ⅱ）と同じ順序で記述する。

（１）（２）（３）ＨＯＳＴからの繰り返し指令を受けたとき、slaveは必ずノードをもつので隣接するＰＥにノードを要求することはない。また隣接するＰＥからノードを要求されることもない。

（４）（Ⅱ）の（４）と同じ。

（５）（４）の結果必ず単一化に成功しノードが生成される。masterが選んだノードと同じノードを選ぶ。

（６）masterから送られてくる指示に従い、individualに変わるかslaveのままであるかを決定する。またindividualに変わる指示とともにノード情報が送られてくれば、これを次に処理すべきノードとする。individualに変わった場合は（Ⅱ）の（７）へ移り、そうでなければ（７）へ進む。

（７）ＨＯＳＴへbusy信号を送る。

（８）（Ⅱ）の（８）と同じ。

４．性能評価

本手法の有効性を評価するため、文献（２）に示されている数式簡単化のプログラムの実行をシミュレーションしてみた。この結果を図４に示す。このプログラムは

？－equiv((2･x¹)･exp(x²)+x²((2･x¹)･exp(x²)),*y).

の質問に対して、式の簡単化を実行し、 2･x¹･(1+x²)･exp(x²)を出力する。なお図４の結果は文献（２）と同様、全解探索を行った結果である。図４の縦軸の加速指数は次式で定義される。

$$\text{加速指数}=\frac{\text{１台のＰＥによる処理時間}}{\text{トーラス型システムによる処理時間}}$$

ここで処理時間は、１個のノードの処理（１個のサブゴール節に対する単一化）に要す時間を単位時間とした値である。またプログラムを実行したときの探索木は、深さ２６でノード数５１３個である。

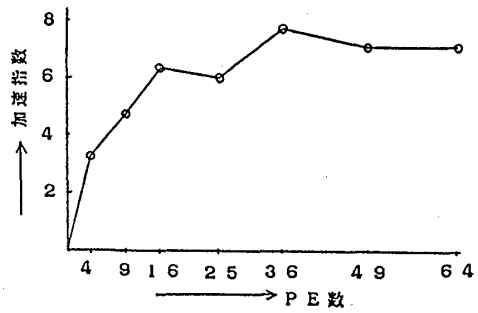


図４．性能評価

５．むすび

本稿では、トーラス型マルチプロセッサシステム上でのPrologプログラムの並列処理手法を提案した。そしてシミュレーション結果から、本手法の有効性が確かめられた（図４における加速指数の飽和現象は探索木が小さすぎるためと考えられる）。本手法では、負荷分散のためのＰＥ間の通信が隣接する４台のＰＥ間でのみ行われ、しかも通信されるノード情報はたかだか２バイトで表現できる。それゆえ、ＰＥの数を増加させても通信時間は変化しない。この点から、本手法は探索空間が大きな場合に対して、特に有効であると考えられる。今後は、ＰＥ数を増加させた場合の本手法の有効性を確認するためのシミュレーション実験を行っていく予定である。

文献

[１] 甲斐，小林，笠原：”階層型挟み打ち探索によるPROLOG ＯＲ並列処理手法”，情報処理論文誌，29,7, pp.647-655(1988).

[２] 相田，田中，元岡：”並列Prolog処理システム”Paralog”について”，情報処理論文誌，vol.24,No.6, pp.830-837(1983).

[３] 川口，真栄田，喜屋武：”トーラス型マルチプロセッサシステム上での分枝限定アルゴリズム”，情報処理研資，マルチメディア通信と分散処理,37-12(1988).