共有メモリ結合マルチプロセッサ上でのKL1処理系のユニフィケーション

2E−11

堂前慶之[1], 今井 明[2], 後藤厚宏[2]
1: (株) 富士通ソーシアルサイエンスラボラトリ
2: (財) 新世代コンピュータ技術開発機構

1　はじめに

ICOTではAND並列論理型言語KL1の高速実行に適した並列推論マシンPIM (Parallel Inference Machine) [1] の開発を行っている。PIMは複数台のクラスタで構成され、各クラスタは共有メモリ/共有バスによって密結合したマルチプロセッサ構成である。ユーザプログラム(ゴール)はクラスタ内の各プロセッサに分散され、プロセッサ毎に独立してスケジューリングされる。分散されたゴール間の非同期通信は、KL1のヒープ中(変数領域)の変数セルへのポインタを共有し(図1)、それをユニフィケーションという操作で具体化することによって行う。従って、KL1の変数セルをアクセスする時には排他制御が必要である。この排他制御にはハードウェアで用意したプリミティブであるCompare & Swap [2] を利用した。

本稿では、Compare & Swapプリミティブを用いて実現したKL1のユニフィケーションの実装例を示す。

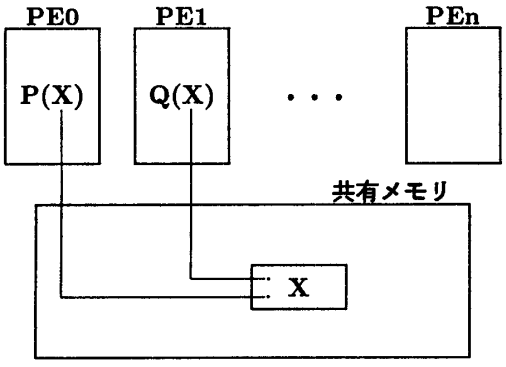

図 1: クラスタ内におけるゴール間の共有変数

2　基本アルゴリズムの設計方針

KL1のユニフィケーションは大部分は代入操作であり、Compare & Swapによる値の代入は変数のロック期間が短いため失敗することは希である。従って、ユニフィケーションの大部分は低コストで効率的に実行することができる。一方、希ではあるがCompare & Swapに失敗した場合、即ち未定義変数を既に他のプロセッサが書き換えていた場合は、そのユニファイの再実行は多少コストがかかっても単純な方式で実現するという方針をとった。

3　ガード部におけるユニフィケーション

KL1のガード部におけるユニフィケーションは具体化した変数の値を参照する操作のため、一般には排他制御は不要である。ただし、対象となった変数がまだ具体化されていない場合には、後に具体化された時にプロセッサの実行時間を浪費することなく効率的にゴールリダクションを再開できるように、ヒープ上の変数セルから具体化を待っているゴールにリンクを張っておく。未定義変数からゴールへリンクを張る時は、未定義変数が既に他のプロセッサによって書き換えられている可能性があるのでCompare & Swapを用いる。Compare & Swapに失敗した場合は、何に書き換えられていたかを調べ、具体化していたらサスペンドさせようとしていたゴールをスケジューリングキューに戻す。

4　ボディ部におけるユニフィケーション

KL1のボディ部におけるユニフィケーションでは、あるプロセッサが未定義変数を具体化しようとする時、既に他のプロセッサによって未定義変数の値が書き換えられている(または、上述のようにゴールにリンクが張られている)可能性がある。従って、Compare & Swapによる未定義変数の書き換えを行う。Compare & Swapに失敗した場合は、そのユニフィケーション再実行のためのゴールを生成することによって単純化することができる。即ち、

```
unify_retry(X, Y) :- true | X = Y.   … (1)
?- unify_retry(A, B).                … (2)
```

(1)のようなプログラムを処理系に組込んでおき、そのプログラムを実行するゴール(2)の第一引数AにCompare & Swapに失敗した変数、第二引数BにCompare & Swap成功時に書き換える値をのせ、通常のユーザゴールと同じスケジューリングキューに入れて再試行する。

KL1 Unification on Shared Memory Multi-Processor
Yoshiyuki DOUMAE [1], Akira IMAI [2], Atsuhiro GOTO [2]
1: Fujitsu Social Science Laboratory Ltd.
2: Institute for New Generation Computer Technology

ユニフィケーション処理はデータの種別によって異なる。以下、それぞれについて述べる。

4.1 アトミック・データ同士のユニフィケーション

2つの引数のタイプと値を比較し、いずれも等しいならばユニファイは成功である。いずれかが等しくないならば、ユニファイは失敗である。KL1は単一代入言語であり、変数は一度値が決まれば変更されないので排他制御は不要である。

4.2 構造体同士のユニフィケーション

2つの引数のタイプと値(構造体本体を指すポインタ)を比較し等しい時、即ち同じ構造体を指しているならばユニファイは成功である。タイプが等しく値が異なる時は、構造体本体の各要素毎のユニフィケーションを行う。ストリングは要素に未定義変数を含まず、整数データからなる構造体なので、要素毎の整数データの比較を行う。リスト、ベクタの様な構造体のユニフィケーションでは、その各要素を再帰的に調べる必要がある。一般にスタックを用いる方法が考えられるが、機械語やマイクロ命令での記述は複雑になりがちである。そこで、各要素のユニフィケーションを再帰的に行うゴールを生成することによって単純化できる。

リストとベクタの各要素について再帰的にユニフィケーションを行うプログラムは次の通りである。

```
list_unifier([X1 | X2], [Y1 | Y2]) :- true |
          X1 = Y1, X2 = Y2.
vect_unifier(0, V1, V2) :- true | true.
vect_unifier(N, V1, V2) :- N > 0 |
          subtract(N, 1, N1),
          vector_element(V1, N1, Elm1, NV1),
          vector_element(V2, N1, Elm2, NV2),
          Elm1 = Elm2,
          vect_unifier(N1, NV1, NV2).
```

専用のスタックを用いた処理方式に比べ、上述のプログラムを実行するゴールを生成する方式では、

- ユーザゴールと同じスケジューリングキューを用いることができるため、特別なスケジューリングを考える必要がない。
- スタックの物理的サイズに仕様が規定されない。

という利点がある。

4.3 未定義変数と具体化した値のユニフィケーション

Compare & Swapを用いて未定義変数へ具体化した値を代入する。Compare & Swapに失敗した場合は、そのユニフィケーション再実行のためのゴールを生成することによって再試行する。Compare & Swapに成功して具体化した未定義変数にゴールがリンクしていた場合、そのゴールをスケジューリングキューに戻す。

4.4 未定義変数同士のユニフィケーション

未定義変数同士のユニフィケーションでは、それらが同一の変数であることを示せるような構造を作る。
並列環境では、2つの未定義変数を同時にユニファイしようとするプロセッサが他にも存在した場合、両者が無作為にリンクを張るとループができてしまう。そこで、未定義変数のアドレスを比較して、
『アドレスの上位のセルから下位のセルへ向けてリンクを張る』
という約束によってこれを解決する。従って、未定義変数同士のユニフィケーションではCompare & Swapによってアドレスの上位のセルを下位のセルへのポインタに書き換える。Compare & Swapが失敗した場合は、そのユニフィケーション再実行のためのゴールを生成することによって再試行する。2つの変数セルを指すポインタが等しい場合はユニファイは成功である。
未定義変数に具体化を待つゴールがリンクしていた場合は、上位のセルにリンクしていたゴールを下位のセルにリンクする。

5 まとめ

本処理方式は、現在PIMシミュレータ上で動作している。本稿では、共有メモリ結合マルチプロセッサにおいて、複雑なユニフィケーション処理をCompare & Swapを用いて単純化した方式を示した。本方式は、処理系のデバッグが複雑化するのを避けるとともに、バストラフィックや共有メモリの競合による性能低下を考慮し、そのオーバーヘッドを必要最低限にするものと考えている。

謝辞

日頃ご指導頂いている瀧 和男室長をはじめとするICOT第一研究室PIM開発グループの方々に感謝致します。

参考文献

[1] A.Goto, M.Sato, K.Nakajima, K.Taki, and A.Matsumoto: "Overview of the Parallel Inference Machine Architecture(PIM)", In *Proccedings of the FGCS'88*,1988

[2] 富田 眞治 末吉 敏則: "並列処理マシン", オーム社,1989