並列論理型言語 Fleng のワークステーションクラスタへの実装

6L－6

大内 敦夫、荒木 拓也、田中 英彦
東京大学工学部

1　はじめに

並列論理型言語は、並列処理における同時実行制御や並列度抽出といった点を取り扱い易い形にしている。本稿では、並列論理型言語の一つである Fleng の処理系の、ワークステーションクラスタ上への実装の際に採用したいくつかの技法について述べる。

2　並列論理型言語 Fleng

並列論理型言語の一種である Fleng のプログラムは、次のような形をしたホーン節から構成されている。

$$H : -B_1, B_2, \cdots, B_n. \ (n \geq 0)$$

Fleng のプログラムの実行は以下のようにして行なわれる。

定義されたホーン節の集合に対して初期ゴールが与えられ、それに対応する定義節を一つだけ選び出す (コミットする)。そして、選んだホーン節のボディ部の各ゴール $B_1, B_2, \cdots, B_n$ に対してまたコミットする定義節を選ぶ。これを、すべてのゴールが処理されるまで繰り返す。

Fleng では単一代入変数やゴールのサスペンド・アクティベートの機構等により、プロセス間の排他制御や同期をあまり意識することなくプログラムを作成することができる。

3　実装に使用した技法

本実装において使用したいくつかの技法について述べる。本実装は、単一プロセッサ用の Fleng 処理系を基に拡張することにより行った。並列論理型言語を並列計算機に実装する際に、単一プロセッサ用の処理系にはなかったいくつかの処理が必要となることが知られており、それらを効率よく行うための技法も既に考案されている。

ワークステーションクラスタ等の分散メモリ型並列計算機の弱点として、通信のオーバヘッドが大きいことが挙げられるため、各技法とも、通信量を如何にして減らすかを要点としている。

3.1　eager transfer

分散メモリ型並列計算機への並列論理型言語の実装では、コミットした定義節のボディ部にあるゴールが他のプロセッサに送られることにより、そのゴールが持つ変数は、他のプロセッサのメモリを指すことになる。よって、ゴールの実行の際には外部メモリへのアクセスが必要となる。

本実装においては、リストのような構造データを読み出して他のプロセッサに渡す時には、その時点で具体化しているデータをすべて渡す、eager transfer と呼ばれる戦略によってデータを転送する。この戦略は、構造データのトップレベルだけを転送する lazy transfer と呼ばれる戦略に比べて全体として速度が速いことが、並列論理型言語の一つである KL1 の処理系 KLIC の分散メモリ環境への実装において示されている [1]。

3.2　外部メモリ参照と分散ガーベジコレクション

本実装においては、プログラムの実行に必要なメモリはセルという単位で確保される。本実装では、各プロセッサの持つ空きセルが不足した時に、ローカルにガーベジコレクションが行われる。この時、他のプロセッサにあるゴールが持つ変数から参照されているようなセルは、ガーベジとして回収することはできない。そこで、輸入表 / 輸出表及び WEC[2] という技法を用いている (図 1参照)。輸入表 / 輸出表はローカルガーベジコレクションが外部参照解決に影響を与えずに行えるようにするために、WEC は輸出表のガーベジコレクションのために用いられる。

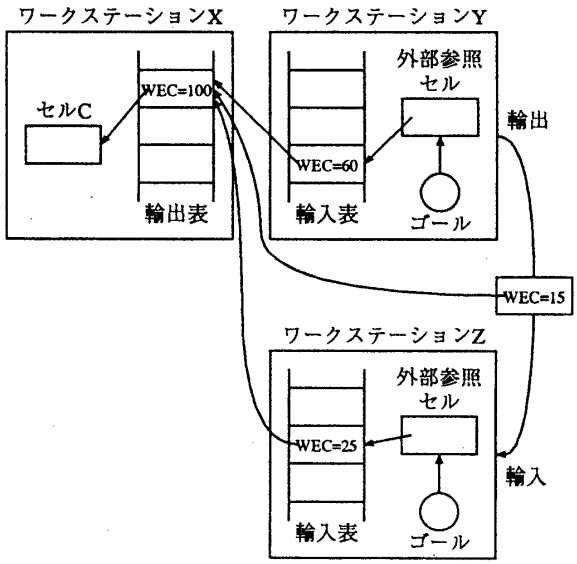

図 1: 輸入表 / 輸出表及び WEC を用いた外部参照

3.3　ヘッドユニフィケーション時の効率化

ゴールに対してコミットする定義節を選ぶ処理はヘッドユニフィケーションと呼ばれる。この時、定義節を選ぶためにはゴールが持つ外部参照を解決しなければならないことがある。ヘッドユニフィケーションは頻繁に行わ

Implementation of Concurrent Logic Programming Language Fleng on Workstation Cluster
Atsuo OUCHI, Takuya ARAKI, Hidehiko TANAKA
Faculty of Engineering, University of Tokyo

れるため、この外部参照解決に必要な通信時間をできる限り無駄にしないようにするべきである。

本実装では、ヘッドユニフィケーション時に外部参照を解決する必要が生じた場合は、輸入表に対してゴールをフックし、返答が来るまでの間に他のゴールのリダクションを行なうようにしている。更に、まだバインドされていなかった場合、輸入表へのフックを残しておくと同時に、セルの輸出元の側で輸出表へのフックを作り、バインドされた時に通知させるようにしている (図2参照)。

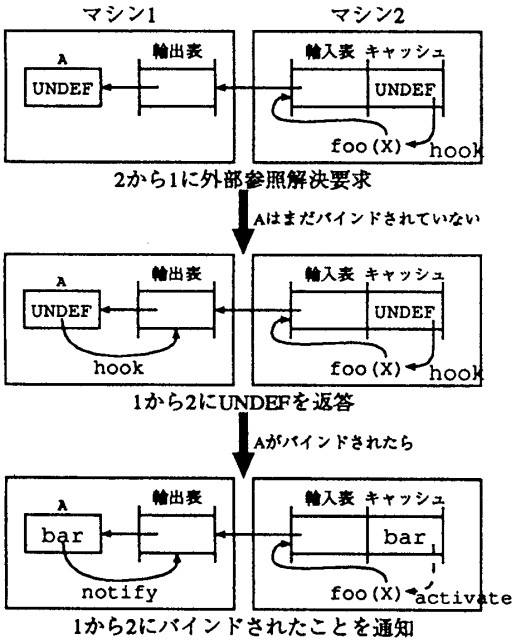

図 2: ヘッドユニフィケーション時の通信中フック構造

3.4 外部参照セルのユニフィケーション

バインドされていない外部参照セルが具体値とユニファイされた場合、外部参照セルの輸出元に対して、バインドするよう通知しなくてはならない (図3参照)。

また、バインドされていない外部参照セル同士をユニファイする場合や、バインドされていない外部参照セルを、バインドされていないローカルなセルとユニファイする場合は、参照のループができることを避けなければならない (図4参照)。現在の実装では、この場合には一旦外部参照を完全に解決し、プロセッサに通し番号を付けておいて、それぞれのセルが存在するプロセッサの番号の大きい方のセルから小さい方のセルを参照するようにしているが、この点に関してはまだ改良の余地があると考えられる。

3.5 負荷分散

本実装では現在のところ、実行時にゴールを他のプロセッサに送出するかどうか決定する動的負荷分散は実装しておらず、プログラム中に明示的に、ゴールをどのプロセッサに送出するかを記述するようにしている。動的負荷分散の実装は今後の課題である。

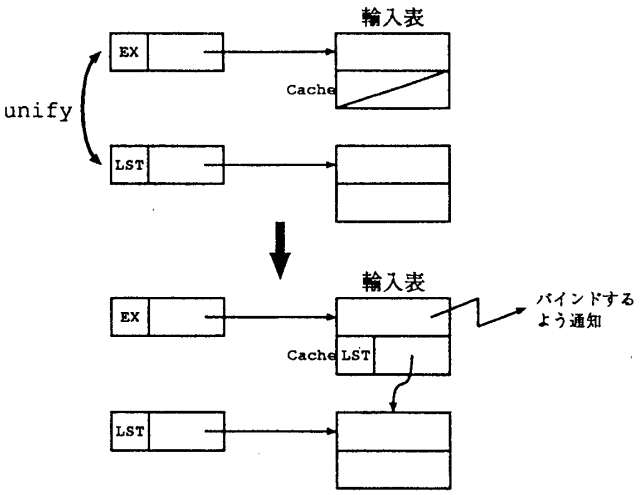

図 3: 具体値と外部参照とのユニフィケーション

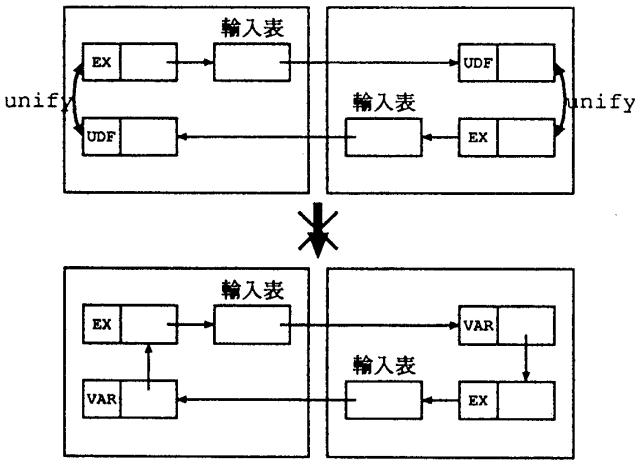

図 4: 外部参照によるループ

4 終わりに

Fleng処理系をワークステーションクラスタ上に実装するにあたり、単一プロセッサ用処理系と比較してどのような処理を加えたかを述べた。今後、実装を終了して初期評価を与えた上で、実行速度の改善や動的負荷分散機能の実装を行う予定である。

参考文献

[1] A. Nakase, K. Rokusawa, T. Fujise and T. Chikayama, "Preliminary Evaluation of a Distributed Implementation of KLIC", ICOT Technical Report TR-0896, ICOT, November 1994.

[2] N. Ichiyoshi, K. Rokusawa, K. Nakajima and Y. Inamura, "A New External Reference Management and Distributed Unification for KL1", In Proc. of the International Conference on Fifth Generation Computer Systems, ICOT, 1988.