並列論理型言語のデータ並列言語による実装

1T－10

古川英司　田中二郎
筑波大学

1 はじめに

最近、各プロセッサにメモリが分散し、それぞれのプロセッサが一つの処理単位を構成するような高並列計算機が、商用のレベルになっている。

しかし、そのような計算機上では、言語処理系として SIMD 的な言語が実装されている場合が多い。SIMD 的な言語でプログラミングを行なう場合、数値計算のような特殊な用途では有効であるが、汎用的なプログラムを記述するには向いていない。一方 MIMD 的な言語を計算機上に直接実装する方法もあるが、実装に要する手間や、他の計算機へのポータビリティを考えると適切な方法とはいえない。

そこで我々は MIMD 的な言語でプログラムを作成しつつ、SIMD 的な言語で実行することによる、汎用的でかつ性能の高いプログラミング手法を考案した。

本研究では、並列論理型言語 GHC[1] のサブセット (MIMD 的な言語) を、高並列計算機 CM-5 上のデータ並列言語 C*[2] (SIMD 的な言語) で実装する方法 [3] について述べる。

2 GHC

並列論理型言語とは、論理型言語 Prolog を並列化したもので、シンタックス的には、すべての定義節はコミットオペレータを持っている。並列論理型言語のプログラムは、定義節の集合として表現され、それぞれの定義節は、ヘッド、ガード (コミットオペレータの前のゴール列)、ボディ (コミットオペレータの後のゴール列) の三つの部分からなる。

並列論理型言語で書かれたプログラムを実行するには、プログラムに対して、あるゴール列 (キュエリ) を与える。キュエリのすべてのゴールは並列に実行 (リダクション) される。リダクションは次のように行なわれる。1. ゴールがユニフィケーションなどの組込み述語であればすぐ実行される 2. ゴールがユーザ定義述語であれば、プログラムの定義節の中からヘッドのユニフィケーションが可能でガードを満足する定義節が探索され、もとのゴールはそのボディのゴール列に展開される (図 1参照)。3. 実行されるべきゴール列が空であれば、実行が終了する。

3 C*

データ並列処理言語とは、多くのデータに対する演算を並列に行うための言語である。C* は、高並列計算のために C を拡張した言語であり、おもに以下のような特徴が追加されている。

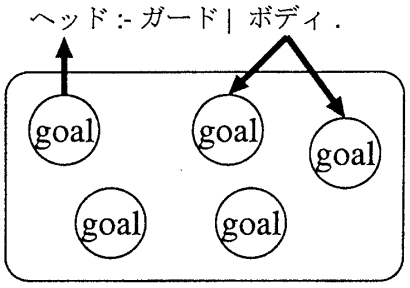


図 1: GHC の実行機構

- 実プロセッサを仮想的に扱うために導入された shape によって、並列データを容易に扱える
- shape を選択する with、並列データを制限して操作する where がある
- 並列データを扱うための、演算子・関数 の拡張がある

例えば、プログラム例を図 2の左に、その実行の様子を図 2の右に示す。

```
1: shape [10]S;
2: int:S p, q;
3: with (S) {
4:     p = 1;
5:     q = p * 3;
6: }
```

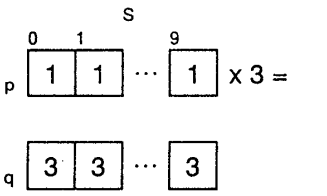


図 2: C* の例

このプログラムでは、1 行目で 10 個の仮想プロセッサを用意し、2 行目で各々の仮想プロセッサ上に int p, q という変数を定義している。3 行目で各々の仮想プロセッサがアクティブになり、4 行目で各々の仮想プロセッサ上の p に1 を代入し、5 行目で各々の仮想プロセッサ上の p を 3 倍してから q に代入している。図 2の右は、5 行目の実行の過程を表している。

4 実装

本研究では、実装の基本となる部分、つまり GHC の並列性の基本部分である **複数のゴールを同時に書き換える** 処理を、C* の特徴の一つである**データの同時処理** によって行なっている。

GHC の一つのゴールは、一つの仮想プロセッサに割り当てる。ゴールが割り当てられれば、仮想プロセッサを活動させ、ゴールの実行がおわれば、仮想プロセッサを休止させるようにする。ボディの展開は、活動中でない仮想プロセッサに割り当てることにする。

Implementation of Parallel Logic Programming Language with Data Parallel Programming Language
Eiji FURUKAWA and Jiro TANAKA
University of Tsukuba

```
goal1 :- true | goal3.
goal2 :-
  true | goal4, goal5.
goal3 :- true | true.
goal4 :- true | true.
goal5 :- true | true.
?- goal1, goal2.
```

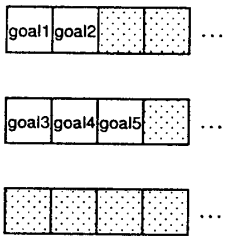


図 3: プログラム実行の様子

図3 は GHC のプログラムを本処理系で実行している過程を示している。左の図がプログラムで、右の図がその様子である。まずキュエリである goal1 と goal2 をゴール列に展開し、それぞれのゴールを並列にリダクションする。すると、ゴール列は goal3 と goal4, goal5 になり、次にそれらのゴールをリダクションすると、ゴール列は空になり、実行が終了する。

本方式では、活動中のゴールはすべて同時に実行され、活動中のゴールがなくなった時プログラムは終了する。また、すべてのゴールの実行が行き詰まった時デッドロックを起こしたとみなしプログラムは異常終了する。

4.1 データ構造

上で述べた基本的な処理を行なうため、データには、

- 複数のゴールから同時に別々のデータをアクセスする
- 条件を満たすゴールの選択を、同時に行なう
- 複数のゴールの書き換えを、同時に行なう

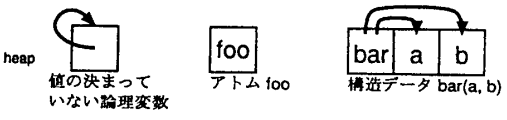
といった機能が必要である。しかし C* には、いくつかの表現上の制限があり、そのおもなものとして、

- 並列構造体はポインタ変数を要素に持てない
- 構造体は並列変数を要素に持てない
- 並列データは並列でない配列の添字になれない

がある。

このため、本実装におけるデータの構造を以下のように決めている (図 4)。

- データは、論理変数や構造データ等の共用体とタグを要素とする並列構造体にする
- データへのポインタは、そのデータがヒープ (並列構造体) の何番めの位置にあるか、その数字 (仮想プロセッサの座標) で表す



データはすべて同時にアクセスできる

図 4: データの実装

データを並列構造体とすることにより、それぞれのデータに対して、同時にアクセスすることが可能になる。また、データをばらばらにしておくのではなく、構造体にしておくことによる若干の性能向上が期待できる。

ゴールがユーザ定義述語である時の定義節の探索や、その定義節が見つかった時のボディの展開も、並列に行なうようにする。そのため、入力された GHC のプログラム (定義節) は並列変数にしている。また、ボディの展開も並列に行なうための並列に並列変数を割り付ける関数や、割り付けられた並列変数にボディを展開するための for 文と where 文、if 文、break 文を組みあわせたループを使っている。

4.2 ユニフィケーション

基本的な処理以外の代表的な処理としてユニフィケーション (ボディ・ユニフィケーション) がある。次のような処理になる。

- それぞれのデータがもし値の決まっている変数ならば、その値をデータとする
- データ同士がが構造データであれば、再帰的にユニフィケーションを適用する
- 一方のデータが変数ならば、もう一方のデータをその変数の値となるようにする。

ユニフィケーションされるデータは並列変数なので、複数のデータを同時に実行することができる。

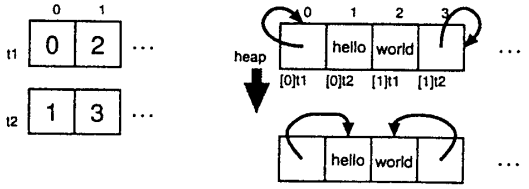


図 5: ユニフィケーションの実行

図5は、並列データ t1 と t2 をユニファイする過程を示している。t1 と t2 はヒープを指している並列ポインタであり、これらをユニフィケーションすると、それぞれの並列データが同時にユニファイされる。

ガード・ユニフィケーション (ヘッドのユニフィケーションを含む) は、上で述べたボディ・ユニフィケーションとほぼ同じであるが、ガード側の変数のみ値を決めることができ、もとのゴール側の変数は値を決めることができない、という点が異なっている。

5 おわりに

本研究では、並列記号処理言語 GHC をデータ並列言語 C* で実装する方法について述べた。現在、その実装法にに基づく処理系を作成しており、コードサイズは C* で 約 1200 行ある。CM-5 上での動作を確認している。今後、本方法の効率化や、それに伴う言語の拡張を行なっていく予定である。

参考文献

[1] K. Ueda: Guarded Horn Clauses, ICOT Technical Report TR-103, ICOT, 1985.

[2] Thinking Machines Corporation: *C* Programming Guide*, May 1993.

[3] 古川英司: 並列記号処理言語のデータ並列言語による実装, 卒業研究論文, 筑波大学, 1994.