7Y-2

密結合マルチプロセッサ上でのFGHC処理系の実現

松田秀雄*, 石田英雄**, 金田悠紀夫*, 前川禎男*
* 神戸大学工学部, ** 松下電器産業

1. はじめに

本研究では密結合マルチプロセッサ Balance 8000 上でFGHCの処理系を作成し，その上でいくつかの評価プログラムを実行して性能評価を行なった．Balance 8000は32ビットマイクロプロセッサNS32032を要素プロセッサとして最大12台までバスにより結合できる．

FGHC (Flat GHC) はGHC[1)]のサブセットであり，ガードにユーザ定義の述語が書けないという制約がある．それに加えて，本研究で実現したFGHC処理系では，実現を簡単にするため以下のような制約を加えている．

・節の呼出しは上から下に逐次的に行なわれる．
・受動部の実行は左から右に逐次的に行なわれる．

2. 処理系の実現

2.1 中間言語命令

本研究で実現した FGHC 処理系では，FGHC のプログラムを以下のようにコンパイルして実行する．

FGHCプログラム → 中間コード → オブジェクトコード

中間コードはCプログラムの形になっている．FGHC のプログラムがどのようにCに変換されるかを図1に示す．FGHC プログラムで同一の述語をヘッドに持つ節が手続きとしてまとめられ，1つの手続きが1つのCの関数にコンパイルされる．手続き内の各節は図1に示すように単純な if 文の連鎖の形にコンパイルされる．これは節の呼出しおよび受動部の実行順序をそれぞれ上から下，左から右に固定したためである．if 文の条件には節の受動部がコンパイルされたユニフィケーションまたは組込み述語の実行を行なう中間言語命令がくる．これらの命令はその実行が成功すれ

```
ヘッド1 :- ガード1 | ボディ1.
          :
ヘッドn :- ガードn | ボディn.
```

(a) FGHC のプログラム（ヘッドの述語が全て同じもの）

```
述語名 ()
{
    if (op (ヘッド1))
    if (op (ガード1))
    {
    op (ボディ1) ;
    return;
    }
      :
      :
    if (op (ヘッドn))
    if (op (ガードn))
    {
    op (ボディn) ;
    return;
    }
    suspend () ;
}
```

(b) 中間コード (op()はヘッド，ガードまたはボディがコンパイルされた中間言語命令を示す)

図1 中間コードの形式

ば真，失敗またはサスペンドすれば偽の値を返すので，失敗またはサスペンドのときは次の節の受動部の実行に移る．全ての節の実行が失敗またはサスペンドしたときにはsuspend() が実行される．この命令で失敗かサスペンドの判断をし，サスペンドのときにはサスペンド処理が行なわれる．

能動部がコンパイルされた中間言語命令は if文の実行部に置かれ，最後はかならず return文で終わる． FGHCの手続き呼出しはCの関数呼出しで実現されているので，戻り番地がスタックに積まれる．return文はこれを解放するためのものである．

Implementing FGHC System on Tight-coupled Multiprocessor
Hideo MATSUDA[1], Hideo ISHIDA[2], Yukio KANEDA[1], Sadao MAEKAWA[1]
1 Kobe University, 2 Matsushita Electric Industrial Co., Ltd.

中間言語命令はICOTのKL1[2)]と同様WAM(Warren Abstract Machine)を基礎としFGHCの実行用に拡張した命令になっている．能動部のユニフィケーションを行なう wait命令，受動部のユニフィケーションを行なう get命令，ゴールの引数に値を設定する put命令，並列実行，サスペンドなどの制御を行なう制御命令および組込み述語命令に分かれる．

2．2　並列実行方式

本処理系での並列実行の手順を以下に述べる．

① ＰＥ（要素プロセッサ）の台数分だけプロセスを生成しＰＥに割り当てる．
② 初期ゴールをレディキューに登録する．
③ ゴールがプロセスによって取り出され実行される．
④ 能動部のゴールが次々にレディキューに登録され，プロセスがそれを取り出して並列実行が進む．
⑤ サスペンドしたプロセスはサスペンド領域に置かれレジュームしたプロセスによりレディキューに移される．

2．3　ガーベジコレクション

本処理系では，処理が単純で特殊なハードウェアの支援を必要としない，再使用可能な領域が全て回収できるなどの理由から印付け法によりガーベジコレクションを行なうことにした．ガーベジコレクションを複数のプロセッサにより並列に行なうことにより処理時間を短縮している．

3．性能評価

本処理系の性能評価をするため，８クィーン，素数生成（1000までの素数），クィックソート（512要素）の３つの問題を実行した．各問題の規模と実行結果を表１に示す．表１で並列ゴールの個数とはレディキューに登録されたゴールの総数である．能動部のゴールのうちで一番左のゴールは逐次的に実行される．表１の並列ゴールの比率は問題の並列性を表わしているが，これからはクィックソートより素数生成の方が台数効果があったことの説明がつかない．そこで，並列実行時のゴール数およびサスペンド数の時間変化を調べた．これらのピークの値を表２に示す．表２からわかるようにプロセッサ台数が大きいとクィックソートのゴール数，サスペンド回数が急激に増大する．クィックソートで台数効果が得られなかったのはこのためであると考えられる．また，ガーベジコレクションについては，８クィーン，素数生成の実行でガーベジコレクションが起動され，それぞれ約70%，約98%の領域が回収できた．その処理時間はプロセッサ１台の８クィーンで約1.4秒，素数生成で約0.43秒なのに対して，８台の８クィーンで約0.22秒（約1/6の時間短縮），素数生成で約0.15秒（約1/3の時間短縮）となった．

4．おわりに

密結合マルチプロセッサでFGHC処理系を実現した．いくつかの評価用問題を実行したところ，プロセッサ10台の８クィーンで8.4倍の実行速度の向上が見られた．また，印付け法を並列に行なうガーベジコレクションを実現しその有効性を示した．

表１　評価用の問題の規模と実行結果

		queen	prime	qsort
ﾘﾀﾞｸｼｮﾝ数		38879	16959	6843
並列ｺﾞｰﾙの個数		11016	169	1025
並列ｺﾞｰﾙの比率(%)		28.3	1.0	15.0
実行時間（秒）	１台	37.05	12.83	5.78
	１０台	4.4	2.77	1.53
実行速度（KRPS）	１台	1.05	1.32	1.18
	１０台	8.84	6.12	4.47

KRPS: Kilo Reductions Per Second

表２　ゴール数とサスペンド回数（ピーク値）

問題	8Queen		Prime		Quick Sort	
ﾌﾟﾛｾｯｻ台数	1台	8台	1台	8台	1台	8台
ｺﾞｰﾙ数	945	982	1	23	131	1806
ｻｽﾍﾟﾝﾄﾞ回数	0	0	0	22	0	321

[参考文献]

1）Ueda,K.: Guarded Horn Clauses,Proc. of the Logic Programming Conference '85 9.3, pp.225-236(1985).

2）Kimura,Y. and Chikayama,T.: An Abstract KL1 machine and its instruction set, Proc. of Symposium on Logic Programming (1987).