並列推論マシン PIM/p への KL1 処理系の実装

畑澤宏善[1], 中越靖行[1], 宮崎芳枝[1], 平野喜芳[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、並列論理型言語KL1を実行するための並列推論マシンPIM(Parallel Inference Machine)の開発を行なっている。PIMにはアーキテクチャの異なる複数のモデルがあり、その一つであるPIM/p向けにKL1処理系の実装を進めている。PIM/p[1]は、8台の要素プロセッサがメモリ共有密結合されたクラスタを、ネットワークで結合する二階層構成をとっている。PIM/pの要素プロセッサは、タグアーキテクチャのプロセッサで、RISCふうの命令セットと、マクロコールと呼ぶ高速のサブルーチン呼び出し機構をサポートしている。

本稿では、先に報告した方式[2]に対し、より効果的なマクロコールの利用法を中心にしてPIM/p向けKL1処理系の実装方式を説明する。

2 マクロコール機構

PIM/pの特徴の一つは、マクロコールと呼ぶサブルーチン呼び出し機構をサポートしていることである。

マクロは、各要素プロセッサの内部命令メモリにおかれる一種のサブルーチンである。内部メモリはキャッシュを介さずにアクセスできるため、キャッシュミスによる命令フェッチの遅延がなく高速に実行できる。

マクロコール命令は、タグ条件分岐と組み合わされており、1命令で条件判定とマクロ呼び出しを実行できる。これによって、KL1処理系に多いデータ型による分岐を高速化できる。

また、マクロコール命令では、最大3個のレジスタ引数を渡すことが出来、マクロ本体からレジスタの内容を参照できるため、スタックを使ったサブルーチン呼び出しに比べ引数受け渡しのコストが小さくなる。

3 抽象機械 VPIM

ICOTでは、各PIMに共通のKL1処理系の仕様となる、抽象機械VPIM (Virtual PIM)を開発した[3, 4]。

VPIMは、PIM仕様記述言語(PSL)で記述されている。PSLは、PIMの機能仕様および実現方式の記述を目的とした、タグアーキテクチャ指向のマクロ展開言語である。また、ツールによってPSLをC言語に変換すれば、VPIMはKL1のインタプリタとして動作する。これを用いて、処理系の実現方式に関するデバッグを汎用計算機上で行なっている。

一方、PIMの各モデル毎の処理系の実装にあたっては、モデル毎のPSLコンパイラ[5]がマシン命令を生成する。ただし、各モデルの特徴に合わせてメモリ使用効率や処理速度の向上を図る部分に関しては、VPIMを書き直すことにした。

4 実装方式

PIM/pのKL1処理系では、VPIMに記述された抽象機械語KL1-Bの処理を、KL1コンパイラでマシン命令に展開する部分と実行時ライブラリとして呼び出す部分に分けて実装した(図1)。

4.1 KL1コンパイラによる展開

KL1コンパイラ[6]は、KL1プログラムを抽象機械語KL1-Bに翻訳した後、展開テンプレートを用いてPIM/pのマシン命令に展開する。

このテンプレートで、各抽象機械語に対する処理のうち、比較的単純なものや、実行頻度の高い部分を、直接マシン命令に展開できる。展開されたコードはマシン命令レベルでの最適化(後述)が期待できる。

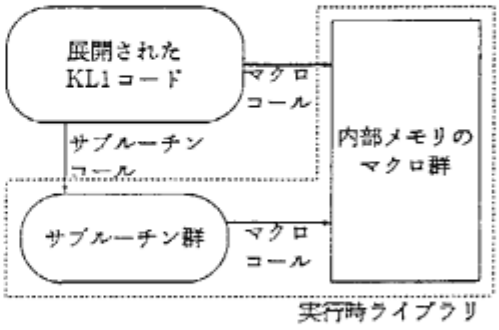

図 1: KL1 処理系の実装形式

Implementation of KL1 Processing System for PIM/p
Hiroyoshi HATAZAWA[1], Yasuyuki NAKAGOSHI[1], Yoshie MIYAZAKI[1], Kiyoshi HIRANO[2]
1: Fujitsu Social Science Laboratory Ltd.
2: Institute for New Generation Computer Technology

4.2 展開部分と実行時ライブラリの切り分け

テンプレートによる展開部分を多くすると、実行速度は向上するが、次のような問題が生じる。

- オブジェクトコード量の増大
- コード量の増大にともなう、命令フェッチ時のキャッシュヒット率の低下

そこで、各抽象機械語の処理で、ある程度複雑なものは、実行時ライブラリとして用意し、テンプレートではサブルーチン呼び出し命令に展開することにした。

処理内容をテンプレートで直接展開するか、サブルーチンを呼び出して実行するかの判断には、展開した際のコードの大きさが決め手になる。具体的な判断基準としては、キャッシュブロックの大きさ(PIM/pでは32バイト)が考えられ、

- 各抽象機械語のテンプレートが、およそ6命令以内(PIM/pの1命令は4または6バイト長)で記述できること

を目安として、展開部分と実行時ライブラリの切り分けを行なった。

4.3 マクロコール命令の利用

実行時ライブラリとして呼び出されるサブルーチンの一部を、マクロ化する、すなわち内部命令メモリに置くことで、マクロコール命令を利用した高速化を図る。

しかし、内部命令メモリは8Kマシン命令分の大きさしかないために、すべてのライブラリをここに置くことは出来ない。どのようなサブルーチンをマクロ化するかの判断材料として、次の二点が考えられる。

- 実行頻度の高いサブルーチンをマクロ化すれば、サブルーチン呼び出し全体のコストを押えられる。
- ある一定以上の大きさのルーチンであれば、マクロ呼び出し時のコストよりも、マクロ化によるキャッシュミスの減少分の方が大きくなるために、インライン展開するよりも高速な実行が期待できる。

これらのことから、デレファレンスルーチンなど、インライン展開できるほど小さくはないが、実行頻度が高いルーチンのマクロ化が有効と考えられる。

これに基づいたサブルーチンのマクロ化は、現状のサブルーチンの実行頻度の評価を元に今後行なっていく。

5 最適化のために

PIM/pにおける、KL1プログラムのより高速な実行を目指して最適化について検討した。

PIM/pのKL1処理系においては、KL1プログラムはKL1コンパイラによって、また、実行時ライブラリはPSLコンパイラによってどちらもPIM/pのマシン命令に展開される。マシン命令レベルでは、一般のコンパイラと同様に、無駄な命令を除去するといった最適化手法が使える。また、PIM/pでは、マシン命令の処理を4段のパイプラインを使って実行しているため、命令の入れ替えや、ディレイド命令の利用などの最適化によってインタロックを防ぐことが出来る。これらの最適化は、KL1プログラム、実行時ライブラリの両者に共通のものであり、また、KL1コンパイラ、PSLコンパイラとも、すべてKL1で記述されていることから、共通の最適化ツールの開発が可能である。

6 まとめ

本稿では、PIM/pにおける、KL1処理系の実装方式について説明し、その最適化について検討した。今後は、KL1プログラム実行時の命令フェッチにおけるキャッシュミスの比率や、各サブルーチンの実行頻度等の測定を行なう予定である。その結果で、マクロ化するべきサブルーチンを検討し、展開テンプレートと実行時ライブラリの切り分けの妥当性を評価して、PIM/p上のKL1処理系の改良を進める。また、最適化についてもKL1コンパイラ、PSLコンパイラに実装し、より高速なKL1処理系の開発を目指す。

謝辞

日頃御指導をいただいている、渥和男室長をはじめとするICOT第1研究室PIM開発グループの皆様に感謝致します。

参考文献

[1] 服部, 他: 並列型推論マシン PIM/p のアーキテクチャ, 情報処理学会論文誌, Vol.30, No.12, pp.1584-1592, 1989.

[2] 宮崎,畑澤,平野: 並列推論マシンPIM/p用KL1処理系の実装, 情処研報 91-ARC-89, pp.209-216, 1991.

[3] 山本, 他: 並列推論マシンPIMにおける抽象機械語KL1-Bの実装 – 高級機械語を実装するための道具立 –, 信学技報 CPSY 89-168, 1989.

[4] ICOT 第 1 研究室編: VPIM 処理方式解説書, TR-1044, ICOT, 1991.

[5] 中越, 他: VPIM及びその開発言語PSLについて, 第39回情報処理学会全国大会 4W-7, 1989.

[6] 平野, 後藤: 並列論理型言語KL1のコンパイル方式の改良, 並列処理シンポジウム JSPP '90 論文集, pp.281-288, 1990.