並列論理型言語 KL1 の静的解析について

2P－5

福田 茂紀　近山 隆
東京大学　工学系研究科
fukuta@logos.t.u-tokyo.ac.jp

1　はじめに

並列論理型言語 KL1 は、処理実行順序や変数の型など、動的に決定しつつ処理を進めていく部分が多い。このためにデバッグが困難になり、また実行時のオーバヘッドも大きくなってしまうという問題があった。

そこで静的解析を行なって、あらかじめ決定可能な情報を抽出し、デバッグや最適化のために用いる研究が進められている。従来の研究として、変数の型やモード (入力か出力か) を制約充足問題に帰結させ、誤りを制約式の矛盾、という形で発見する手法 [5] などがある。

しかし従来の KL1 における静的解析はデバッグのための解析が中心で、プログラムの意味を変えずに効率的に動作させるために付加されるプラグマの情報に、注目していなかった。

今回は、並列動作指定を行なうプラグマ情報にも着目した KL1 の静的解析手法についての研究を行ない、特に通信部分のボトルネックとなりやすい通信回数の増加を押さえるための最適化手法を提案し、その効果を確かめるための予備実験を行なった。

2　並列論理型言語 KL1

2.1　KL1 における変数

手続き型言語において、変数とはメモリ領域につけた名前である。このため、変数の値を計算途中で変更していくことができる。

KL1 において変数とは値そのものに付けた名前である。この変数は「未定」と「既定」の二つの状態を持ち、一度決まった変数がその値を変えることはない。KL1 において、変数は英大文字で始まる英数字列で表現される。

2.2　KL1 における手続き

KL1 における手続きは、「手続き (引数列) :- 手続き (引数列), 手続き (引数列).」といった形で記述される。先頭の手続きが呼び出された時には、" :- " 以下の処理が行なわれる、という意味である。

今後、論理型言語の通例にしたがって、各手続きを「述語」と呼ぶことにする。また、述語の名前は同じでも、引数が違うと別の手続きとみなされるため、各述語は「名前 /引数の数」と表記される。

2.3　並列実行の指定

KL1 は並列処理言語である。つまり、全体の処理の一部を別の処理単位に計算させ、結果を待つ間に別の処理を進めておくことができる。

この別の処理単位を "ノード" と呼び、個別のプロセッサ、または内部では自動負荷分散するようなプロセッサの集まりである。

ある述語 Goal をノード Node で実行させたい場合には、以下のように表記する。

$$Goal@node(Node)$$

この $@node(Node)$ 部分をゴール分散プラグマと呼ぶ。

3　ノード間通信最適化手法について

3.1　KL1 における通信

KL1 によるプログラムでは、構造化されたデータを多用し、この構造の深さは非常に大きくになることがある。

このため KL1 の処理系 KLIC では、通常はデータ送信の仕方として、各データが必要になったときに 1 段ずつ送信を行ない、内部構造のデータ内容は、それらを参照する際に改めて通信を行なうことにしている。

しかし、ある程度以上の大きさとなった構造データを別ノードに送信するときは、送信回数が多くなってしまう欠点がある。この送信の度に必要となるオーバヘッドを減らすために、KL1 の処理系は送信を行なうデータの内部構造のうち、すでに計算が終わって完成しているデータはすべて送るモードを用意している。このモードでは、構造内のどのデータが必要であるかを考慮しないため、必要のないデータまで送信してしまい、通信量が多くなるという問題がある。

3.2　予備実験

今回はこの構造体データの送信に関する最適化の効果を計るため、予備的な実験を行なった。

その実験内容は、

- リスト構造は、従来通り 1 レベルずつ送信を行なう
- その他の構造は、完成しているデータをすべて送るという送信ポリシーの元、いくつかのプログラムの実行時間を通常の 1 レベルずつ送信を行なうもの、全レベル送信するものと比較した。今回実験に用いたプログラム

Static analysis of KL1 programs
Shigeki FUKUTA and Takashi CHIKAYAMA
School of Engineering, The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8656, JAPAN

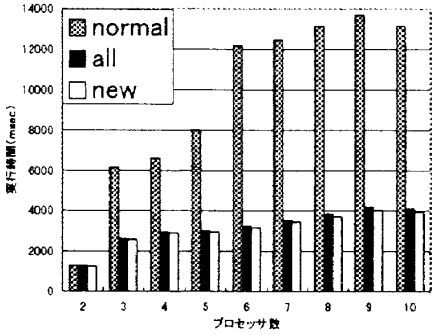

図 1: 例題 1 / ソート

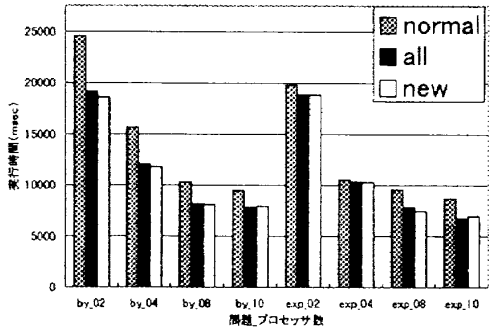

図 2: 例題 2 / ライフゲーム

は、プロセッサ間通信において「1 プロセッサで用いられるデータ」にはリスト構造を用いなかった。

一つは、一定の深さの二分木の葉にデータを作成し、それらを別ノードでソートするプログラムである。一つのノードでソートするデータは一つの構造として送信され、ノード毎の計算量はノード数によらず一定になるようにデータを作成している。こちらは、一度に送信するデータが多い例である。

もう一つは、コンウェイのライフゲームの計算を y 軸方向に分割して並列に行なうプログラムである。このプログラムは、計算領域の境界線上のデータのみをやりとりして計算を行なっている。こちらはノード数が増えても総計算量はほぼ変えない。こちらは、一度に送信するデータが少ない例である。

3.3 実験環境

テストを行なったプログラムのうち二つのプログラムの結果を、以下に示す。

なお、テストは計算機は Sun Ultra Enterprize E3000、OS は Soralis 2.5 で行なった。並列動作は共有メモリ計算機上でメッセージパッシングを行なっている。

3.4 実験結果

実験結果を、図 1と図 2に示す。どちらも今回提案した新手法は、通常の送信を行なった場合に比して計算時間が短縮している。さらに、多くの例で全レベル送信を行なった場合と比べても若干の速度向上が見られる。

また、全レベル送信を行なった場合には送信バッファを大きくとる必要があったが、新手法ではそのような必要はなかった。

4 静的解析による適用

以上のように、予備的実験では計算時間の短縮に成功し、従来の全レベル送信に比べても、送信バッファを小さくすることができた。

今回の実験では、リスト構造のみを 1 レベルずつ送信し、他の構造は全レベルを送信している。また、プログラムもその構成に適したものを用いた。今後は静的に送信先ノードで消費されることが部分構造を解析し、その情報に応じて送信を行なう処理系を実現していく。

参考文献

[1] Takashi Chikayama: KLIC: A Portable Parallel Implementation of a Concurrent Logic Programming Language. Proc. International Workshop on Parallel Symbolic Languages and Systems, Lecture Notes in Computer Science 1068, Springer-Verlag, Berlin, Oct 1995, pp.286-294.

[2] 長 健太, 上田 和紀: モード誤りをもつ並行論理プログラムの静的デバッグ手法. 1996 年並列処理シンポジウム論文集, 情報処理学会, 1996 年 6 月, pp. 219–226.

[3] Kenta Cho, Kazunori Ueda: Diagnosing Non-Well-Moded Concurrent Logic Programs. In Proc. 1996 Joint International Conference and Symposium on Logic Programming (JICSLP'96), M. Maher (ed.), The MIT Press, 1996, pp.215-229.

[4] Kazunori Ueda: Experiences with Strong Moding in Concurrent Logic / Constraint Programming. Proc. International Workshop on Parallel Symbolic Languages and Systems, Lecture Notes in Computer Science 1068, Springer-Verlag, Berlin, April 1996, pp.134-153.

[5] 上田 和紀: 平成 9 年度　委託研究ソフトウェアの最終成果報告書 (9) KL1 プログラム静的解析系. http://www.icot.or.jp/AITEC/FGCS/funding/97/saishuu09.html

[6] 関田 大吾: Inside KLIC. http://www.icot.or.jp/AITEC/COLUMN/KLIC/inside/master.html