ポータブル KL1 処理系 KLIC の概要

3D−7

仲瀬 明彦、藤瀬 哲朗、近山 隆
(財) 新世代コンピュータ技術開発機構

1 はじめに

プログラミング言語 KL1[1] は FGCS プロジェクトの核言語であり, 並列推論マシン [2] 上の OS である PIMOS[3] や応用プログラムを記述することで高並列処理への有効性を実証してきた.

しかしながら, KL1 で記述された数多くのプログラムは, 並列推論マシン上でのみ動作するものが多い. そこで第五世代コンピュータの研究基盤化の一環として, KL1 により実現されているプログラム動作環境を汎用計算機上に実現する. 実現にあたり本環境は, プログラム開発のためのシステムであると同時に開発したプログラムを走らせるための環境となることを強調したい. そのため本環境は並列推論マシンを目標に, 問題によってはそれ以上の性能をもち, 同時に高い移植性を保つことを必要とする.

これらの問題を解決するために KL1 プログラムを プログラミング言語 C で記述されたプログラムに変換する方式の処理系 (以下 KLIC と呼ぶ) を開発中である. 本稿では KLIC の概略について説明し, 試験的に実装した逐次版の性能評価結果についても示す.

2 KLIC の開発方針

KLIC では以下の技術を利用して開発を進める.

- 汎用計算機技術
  KLIC が搭載されるのはワークステーション, 並列 UNIX システム, それらを汎用ネットワーク結合したもの等の汎用計算機システムである. これらのマシンの多くが Unix もしくは Unix をベースとした OS をもち, C を中心とした開発環境を抱えている. このように、ハードウェアを隠蔽するプラットホームとして Unix/C を利用することは、移植性の面から見て大変有利である。また, 優れた最適化を行なう C コンパイラがある等, マシン依存の低レベルな最適化は汎用計算機技術に頼れることも重要である.

A portable KL1 language processor KLIC
Akihiko Nakase, Tetsuro Fujise, Takashi Chikayama
ICOT, Mita Kokusai Bldg.21F, 4-28 Mita 1-chome, Minato-ku, Tokyo 108 Japan

- 並列推論技術
  並列推論マシン上の KL1 の実装技術、並列処理プログラム開発環境さらに応用ソフトウェアは, そのまま汎用計算機上の KL1 処理系で使用できる技術も少なくない. 特に分散データ管理・分散実行管理技術や並列処理プログラム開発環境のデザインが生かせることは大変有利である。
- 汎用技術と並列推論技術の親和性を高める技術
  KL1 の C 言語への効率的変換方式や Unix 環境下での効率的並列分散方式等, 上述の 2 つの技術を結び付ける技術を開発する必要がある. KLIC では, 特に静的解析を中心とした並列論理型プログラミング言語の解析技術 [4], [5] により最適化を行なうことを重要視する.

3 実装の基本方式

KLIC では, 図 1 に示す通り KL1 プログラムを C プログラムに変換する基本方式を採用した. 実行手順例を以下に挙げる.

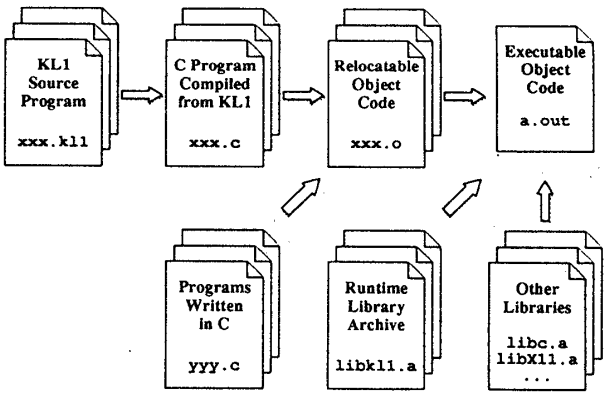

図 1: 実装の基本方式

1. KL1 プログラムを C プログラムに変換
2. C に変換されたプログラムを C コンパイラでコンパイル
3. オブジェクトと実行時ライブラリをリンクし, 実行形式を作成
4. この実行形式オブジェクトを普通の Unix のプロセスとして実行

Lisp や Prolog 等のようにデバッガその他すべての機能を満載した実行環境を提供する方式ではなく，KLIC では実行される KL1 プログラムのみを C へ変換する環境を提供する．

一般に機械語への直接の変換より C に変換する方が，実行効率面で劣ることが予想される．しかしながら以下の面でメリットがある．

- 多様なシステムへ簡単に移植できる．
- 低レベル，マシン依存の最適化は C コンパイラに任せられる．
- 既存プログラムと容易にリンクして使える．

特に他言語プログラムとのリンク方式を提供することにより，KL1 の言語システムの自由な拡張や，他言語プログラムのプロセス群の KL1 による並列制御が可能になる．

4 試験的実装の性能評価

性能上鍵となる機能を中心に，一部の仕様のみ実現した逐次版試験実装により性能評価を行なった．naive reverse (30 要素,1000 回繰り返し) の結果を表 1 に示す．

表 1: 試験的実装の評価 (その 1)

System	Speed	Code Size
Symmetry S81	118 KLIPS	608 bytes
Sun-3/260	205 KLIPS	532 bytes
SparcStation 2	985 KLIPS	640 bytes
SparcStation 10/30	2,000 KLIPS	640 bytes
SparcCenter 2000	2,265 KLIPS	680 bytes
DEC alpha 7000	3,701 KLIPS	928 bytes
SICStus (compactcode)	482 KLIPS	320 bytes
SICStus (fastcode)	1,053 KLIPS	736 bytes

(KLIC: GCC -O2 オプション, GC 時間含む)
(SICStus Prolog: ver.2.1 on SS 10/30, GC 時間含まず)

速度性能に関してもまずまずの結果を得ている．また速度向上度に関して汎用計算機搭載の効果がみられ，また C 言語を中間言語とすることで核部分の高い移植性も確認できた．

次に，論理型言語で使用される他のベンチマークプログラムについて，fastcode でコンパイルされた SICStus Prolog プログラム (直接機械語が生成される) と比較した結果を表 2に示す．実行は SparcStation 10/30 上で行なった．

速度性能では，SICStus Prolog の 1.4 倍から 2 倍の性能が出ている．

コードサイズについては，通常のリスト処理のみを使うベンチマークについては SICStus Prolog と比較しても妥当と思われる．しかし，算術演算やガード部のインデキシングにおいて，若干最適化できてない部分があり，それらの処理を多く含んでいるベンチマークについては，SICStus Prolog より大きなコードが生成されてしまう．これらの最適化は順次行なってゆく予定である．

表 2: 試験的実装の評価 (その 2)

	実行時間		コードサイズ (bytes)		
program	KLIC	SICStus	KLIC	SICStus	S/K
qsort	1	1.41	1160	720	0.62
hanoi	1	1.43	504	496	0.98
prime	1	1.66	1488	768	0.52
lqueen	1	1.84	4344	1088	0.25
mgtp	1	1.68	24648	8448	0.34

5 まとめと課題

KLIC の開発方針と、暫定版処理系の評価について述べた。

KLIC と並列推論マシンとの開発方針に関する大きな違いは以下のことである．

並列推論マシン: 動的処理の効率化に重点
KLIC: 静的解析技術による効率化に重点

KLIC では並列推論マシンと比較してプロセッサが速くなったため，相対的にメモリは遅くなっている．このため，頻繁に利用するワーキングセットを小さくし，キャッシュヒット率を向上させるために generation GC を検討中である．

また KLIC の並列処理に於いては，並列推論マシンとは異なり専用ハードウェアや専用 OS をもたないため，プロセッサ間の通信性能，特にレスポンスが極端に悪くなることが予想される．このことに対処するためには，バッファリングやプログラム解析等による多レベル一括転送機能を実現し，通信頻度を低減する必要がある．

参考文献

[1] Ueda, K. et al.: Design of the kernel language for the parallel inference machine, *The Computer Journal*, December 1990.

[2] Goto, A. et al.: Overview of the parallel inference machine architecture (PIM), *Proceedings of FGCS'88.*

[3] Chikayama, T. et al.: Overview of the parallel inference machine operating system (PIMOS), In *Proceedings of FGCS'88.*

[4] Ueda, K. et al.: Moded Flat GHC and Its Message-Oriented Implementation Technique, submitted to New Generation Computing, 1993.

[5] Sastry, A. et al.: A Compile-Time Memory-Reuse Scheme for Concurrent Logic Programs, TR CIS-TR-91-24 of Univ. of Oregon, 1992.