リフレクション機構を持ったKL1による動的負荷分散*

2J－4

高橋 俊行　武田 正之†
東京理科大学大学院 理工学研究科 情報科学専攻‡

1 はじめに

ソフトウェアの構成原理としてのリフレクションの有用性に着目し、並列論理型言語KL1へリフレクション機構を導入した、KL1にこの拡張を加えることにより、メタレベルコードのモジュール化が可能となり、KL1プログラムの記述しやすさが向上した。導入したリフレクション機構では、リフレクションをボディ実行部に制限したため、メタレベルとベースレベルに分割して記述したプログラムを、これらを合成したKL1プログラムへコンパイルすることができ、効率の良い実行が可能である。本発表では動的負荷分散プログラムを例に、このリフレクション機構の有用性を示す。

2 負荷分散とリフレクション

一般に並列ソフトウェアは問題解法部分と負荷分散部分とから構成されているものと考えることが出来る。KL1がその能力を発揮する大規模な知識処理アプリケーションにおいては、負荷分散方式がアプリケーション全体の性能を左右するため、負荷分散方式の設計・実装は重要な意味を持つ。

第五世代計算機プロジェクトではKL1アプリケーションのために負荷分散ライブラリが作成されたが、この際KL1の言語定義の枠内で考え得る次の負荷分散部分の分離方式が検討された[1]。

- 付加方式
- プロセッサ番号サーバ方式
- 負荷分散テンプレート方式

これらの方式に共通の欠点は「負荷分散方式を選ぶと問題解法の大枠が決ってしまい、柔軟性に欠けること」である。

一方、リフレクション[2]は元来、計算システムが自己の構成や計算過程に関する計算を行なうことであるが、本来行なうべき計算(ベースレベルの計算)と、その計算方式について制御あるいはカスタマイズする計算(メタレベルの計算)の分離を助けることから、ソフトウェアの構成原理として非常に注目されている。

我々はKL1へリフレクション機構を導入し、リフレクションが提供するベースレベルとメタレベルの分離という観点からのモジュール化手法を、KL1ソフトウェア生産性向上に役立てる研究を進めてきた[3][4]。並列ソフトウェアの問題解法部分はベースレベルの計算、負荷分散部分はメタレベルの計算である。我々が拡張するKL1はこれらを独立したモジュールに分離し、柔軟な負荷分散戦略を可能にする。

3 KL1言語の拡張

我々は、メタボディとメタプロセスグループという２つの概念から成るリフレクション機構をKL1へ導入した。

3.1 メタボディ

メタボディはボディ部の実行を制御／カスタマイズするメタレベルの記述である。本リフレクション機構ではメタレベルのプログラムはメタボディへ記述し、ベースレベルから隠蔽する。ベースレベルのボディ部の実行は、指定されたメタボディによってカスタマイズされる。メタボディは拡張されたKL1で記述するが、ベースレベルのボディ部に現われる変数の参照／具体化／置き換えや、ゴールの追加／削除／置き換えによってリフレクション操作を行なう。メタボディの実行をサスペンドすることによって、ベースレベルのボディ実行を延期することも可

*Dynamic load balancing based on Reflective KL1
†Toshiyuki TAKAHASHI and Masayuki TAKEDA
‡Science University of Tokyo

能である。また、従来プラグマで指定していたボディゴールのプライオリティやノードはメタレベルの操作として、メタボディ内で指定する。

3.2 メタプロセスグループ

メタボディの実行をメタプロセスと呼ぶ。メタプロセスグループはメタプロセス間で共有資源を扱うために導入された概念である。メタプロセスグループに属するメタプロセス群はグループコントローラにより統括される。共有資源の管理は、メタプロセスが資源の確保や開放についてグループコントローラと通信することで実現する。

4 動的負荷分散

リフレクション機構を持ったKL1において負荷分散、とくに動的負荷分散が如何に記述されるかを本節では紹介する。

取り上げる負荷分散方式は動的負荷分散の基本系である。あらかじめグループコントローラとして負荷分散管理プロセスを起動しておき、ベースレベルの計算プロセスが分散処理を行なう必要が発生した時に、これを監視していたメタプロセスが負荷分散管理プロセスへ、適切な分散先のノード番号を問い合わせる。そして、メタプロセスは負荷分散管理プロセスが返答したノード番号のもとへベースレベルの計算プロセスを分散させる。

この動的負荷分散を行なう為のメタモジュール(=メタボディおよびグループコントローラの定義モジュール)を以下に示す。

```
:-metamodule msample.
:-metagrp initgrp/1.
initgrp(*strm) :- loadb:demander(*strm).
:-metabody assign_node/1.
assign_node(*strm) :- *strm<<N,
                      $execbody@node(N).
```

:-metagrp 宣言はメタプロセスグループ宣言で、:-metabody 宣言はメタボディ宣言である。メタモジュール中の*strmは、グループストリームを表している。グループストリームはメタプロセスとグループコントローラの間の通信路である。

ベースレベルのプログラムを以下に示す。

```
:-module sample.
%defgrp msample:initgrp(ns).
main :- gen(X), foo(X).
%reflect msample:assign_node(ns).
foo([X|L]) :- calc(X), foo(L).
foo([]) :- true.
```

%defgrpはメタプロセスグループの使用宣言である。また、%reflectは続く節とメタボディとを関連付けさせる宣言である。

コンパイル後のプログラムを以下に示す。

```
:-module sample.
main :- gen(X), foo(X,Ns),
    merge(Ns,Ns1), loadb:demander(Ns1).
foo([X|L],Ns) :- Ns={Ns1,Ns2}, Ns1=[N],
                 foo_b(X,L,Ns2)@node(N).
foo([],Ns) :- Ns=[].
foo_b(X,L,Ns) :- calc(X), foo(L,Ns).
```

5 おわりに

KL1へ導入したリフレクション機構は、アプリケーションのベースレベル、メタレベルといった観点からのモジュール化を可能にし、KL1のソフトウェア生産性を向上させる。導入したリフレクションは効率化のためにリフレクションの範囲が制限されているが、動的負荷分散などの記述は十分可能である。

参考文献

[1] ICOT: *KL1*負荷分散ライブラリ説明書 *(第 1.0 版)*, 1993.

[2] Maes,P.: *Issues In Computational Reflection*, In Meta-Level Architectures and Reflection, pp.21-35, North-holland, 1988.

[3] Takahashi,T. and Takeda,M.: *An Efficient Implementation of Reflection in KL1*, In FGCS'94 Workshop on Parallel Logic Programming, ICOT, 1994, pp.17-26.

[4] 高橋俊行: 並列論理型言語 *KL1* におけるリフレクション機構の研究, 修士論文, 東京理科大学大学院理工学研究科情報科学専攻, 1995.