# KLIC処理系の分散メモリ実装方式

1T－8

六沢 一昭　仲瀬 明彦　近山 隆
(財) 新世代コンピュータ技術開発機構

## 1 はじめに

並列論理型言語KL1[1] は, これまで Multi-PSI, PIM などの記号処理向き分散メモリ型並列計算機に処理系が実装されてきた[2]. これらの処理系は計算機の特徴と密接に関係しており, 性能は高いが移植性は低い.

一方, KL1 を C に変換することによって高い移植性と性能を実現することを目的とした KLIC 処理系[3] が設計された. 構築された KLIC の逐次処理系核は他の類似処理系と比べて高い性能を示している[3, 4].

本稿では KLIC の分散メモリ実装方式を述べる. 分散環境に係わるデータや処理は KLIC の **generic object** という拡張機能を用いて実現した.

## 2 設計方針

高い移植性と性能を得るため以下の方針を立てた.

- Multi-PSI, PIMs における実装方式[5] は充分実績があるので, 基本的にこれを踏襲する.
- KLIC 処理系核 (以下, **カーネル**と呼ぶ) を変更することなくそのまま使う.
- 分散実装に必要なデータ構造やそれに係わる処理は generic object を使って実現する.

## 3 generic object

generic object は新しいデータ型とそれに係わる操作を定義するものである. カーネルは, generic object というデータ型のあることだけを知っており, 操作を定義しているメソッドを必要に応じて呼出す. 実際に定義されているデータの形式や操作の内容は関知しない. このため, generic object を使うと, カーネルに手を加えることなくシステムを変更/拡張することができる.

KLIC には以下の 3 種類のオブジェクトがある.

**data object**　(いわゆる) 新しいデータ型を作るためのオブジェクト. KL1 言語仕様には含まれないデータ型, 例えば多倍長整数はこれによって実現している.

**consumer object**　プロセス形式のオブジェクトであり, 中断ゴールと同じように振舞う[1]. 変数にフックし, 変数が具体化されると起動される (*unify* メソッドが呼ばれる).

データ駆動で起動されるプロセスを consumer object によって実現することができる. ストリームマージャはこれを用いて実現した.

**generator object**　これもプロセス形式のオブジェクトである. consumer とは異なり, 値の生成要求によって起動される (*generate* メソッドが呼ばれる)[2]. *generate* メソッドは「直ちに値を生成する」か「値の生成を行なうプロセスをフォークする」のいずれかを行なう.

要求駆動で起動されるプロセスを generator object によって実現することができる. 優先度付きストリームマージャはこれを用いて実現した.

### ユニフィケーション

**consumer 同士**　二つの変数を単一化し, フックしている consumer (群) はひとつにまとめる.

**consumer と generator**　generator の *generate* メソッドを呼出し, その結果と consumer のユニフィケーションを行なう.

**generator 同士**　いずれかで *unify* メソッドが定義されているならばそれを呼出す. 両 generator とも未定義ならば, いずれか一方の *generate* メソッドを呼出し, その結果ともう一方の generator のユニフィケーションを行なう.

## 4 外部参照ポインタの実現

他のプロセッサを指すポインタを**外部参照ポインタ**と呼ぶ. 外部参照ポインタは generic object で実現した.

ポインタに対しては「参照値の読出し (dereference)」と「ユニフィケーション」を行なうことがある. 前者はオブジェクトに対する値の生成要求と考えられるので *generate* メソッドで実現し, 後者は *unify* メソッドで実現した. ポインタの生成時は二つの処理とも行ないうるので generator で実現するが, 読出し処理が起動される (*generate* メソッドが呼ばれる) と consumer に変わる. もはや読出し処理は行なわないからである.

外部参照ポインタを表わすオブジェクト (generator, consumer) を, 以下では, それぞれ **generator exref object, consumer exref object** と呼ぶ.

---

Distributed Memory Implementation of KLIC
Kazuaki ROKUSAWA, Akihiko NAKASE, Takashi CHIKAYAMA
Institute for New Generation Computer Technology

[1] 中断ゴールは「ゴール用に最適化した consumer object」と考えることもできる.

[2] *unify* メソッドを持ちデータ駆動でも起動される generator を定義することもできる. 外部参照ポインタはその例である.

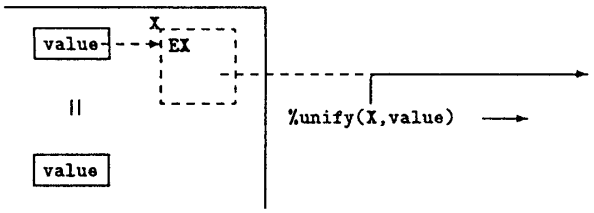


図 1: exref object X と具体値

## 5 分散ユニフィケーション

外部参照ポインタに係わるユニフィケーションを以下に示す. カーネルは, 前述した「generic object に係わるユニフィケーション処理」を行ない, その結果として分散ユニフィケーションが行なわれる.

**exref object X と具体値 value** Xがフックしている変数に具体値を書込み *unify* メソッドを呼出す. メソッドの実行により, X の指すプロセッサへ「ユニフィケーションを依頼するメッセージ(%unify(X,value))」が送られ, X は解放される(図1).

**generator exref object X と consumer (群)** まず X の *generate* メソッドを呼出す. メソッドの実行により, 参照値の読出しを依頼するメッセージ(%read)が送信され, X は consumer に変わる. 次に consumer 同士のユニフィケーションを行なう(図2).

**generator exref object 同士(X,Y)** 一方の generator (X とする) の *unify* メソッドを呼出す. メソッドの実行により, X が指すプロセッサへ %unify(X,Y) が送られ, 「X がフックしている変数」に「Y がフックしている変数へのポインタ」が書込まれ, X は解放される.

**consumer exref object 同士** 二つの変数を単一化し, 二つの consumer(群) をひとつにまとめる. メソッドの呼出しは行なわない. カーネルの処理だけで完結する.

**exref object X と変数 V** Xがフックしている変数へのポインタを V に書込む. この処理もカーネルの処理だけで完結する.

## 6 参照値の読出し(dereference)

generator X の *generate* メソッドが呼ばれると, X が指すプロセッサへ %read(X,Ret) が送られ, X は consumer に変わる. Ret は参照値の返信先である.

%read を受信したプロセッサは, **参照先が具体化されていれば** 「参照値を運ぶメッセージ(%answer)」を直ちに返信する. **参照先が変数**[3]の場合は, 返信先を記録した consumer object (**reply object** と呼ぶ)を生成し変数にフックする. 変数が具体化されると reply object の *unify* メソッドが呼ばれ, %answer が返信される. **参照先が generator exref object X'** の場合は, X' が指すプロセッサへ %read(X',Ret) を転送する.

[3]consumer や中断ゴールがフックしている変数も含む.

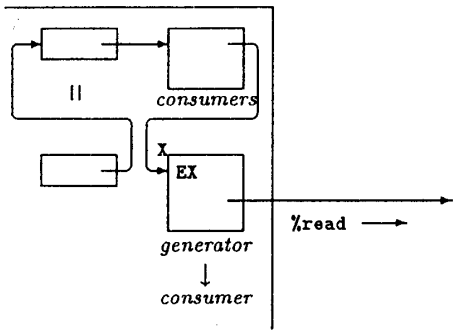


図 2: generator exref object X と consumer (群)

### ループへの対処

KLIC 処理系核は, 変数と『何か』のユニフィケーションを, 「『何か』へのポインタを単純に変数に書込む」ことによって行なう. このためプロセッサ間に渡るループの生まれる可能性がある. generator exref object のみからなるループが存在した場合, 単純に %read を転送すると処理がいつまでも終わらなくなってしまう.

この問題は, %read に**転送可能カウンタ**を持たせることで解決した. カウンタ値が 1 以上ならば −1 して転送する. 0 ならば転送は行なわず *generate* メソッドを呼出し, reply object を生成してフックする. これにより, ループに %read が飛び込んでも, reply object のフックした consumer がループ中に生まれることが保証できる. その結果, 再び %read が転送されてくると reply object のところで転送は終了する.

## 7 おわりに

KLIC の分散メモリ実装方式について述べた. KLIC の拡張機能である generic object を利用し, カーネルに手を加えることなく実現している.

## 参考文献

[1] K. Ueda *et al.* : *Design of the Kernel Language for the Parallel Inference Machine,* The Computer Journal, Vol.33, No.6, pp.494–500, 1990.

[2] K. Taki : *Parallel Inference Machine PIM,* FGCS'92, pp.50–72, 1992.

[3] T. Chikayama *et al.* : *A Portable and Efficient Implementation of KL1,* PLILP'94, 1994.

[4] 近山 隆 他 : KLIC 処理系核の評価, 情報処理学会第49回全国大会, 1T-6, 1994.

[5] N. Ichiyoshi *et al.* : *A New External Reference Management and Distributed Unification for KL1,* New Generation Computing, pp.159–177, 1990.