# 5U-2 論理型言語向き並列計算機KPRのストリーム並列処理方式

川倉康嗣、柴山　潔、萩原　宏
(京都大学・工学部)

## 1. はじめに

KPRは、論理型言語プログラム(論理プログラム)の実行モデルとして我々が提案した「並列リダクション・モデル」に基づいて設計されたマルチプロセッサ・システムである。

本稿では、論理プログラムがKPR上でどのように処理(コンパイルと実行)されるかを示すとともに、論理型言語向き計算機として備えている処理機能について述べる。

## 2. 並列リダクション・モデル

### 2.1 プロセス割り付け戦略

KPRの実行モデルでは、AND/OR木の論理的AND関係にあるノードにはStream(S)プロセスを、OR関係にあるノードにはOr(O)プロセスを割り付ける。Sプロセスは、ストリーム並列処理方式で本体ゴール列を処理し、疑似的にAND並列性を実現する。

KPRにおける論理プログラムの並列処理は、プロセス個々の処理とプロセス間通信処理とから成る。原則的には、プロセスはSプロセスとOプロセスとが交互に起動され、プロセス間通信は親子関係にあるプロセス間のみで行われる。通信メッセージには、次の3種類がある。

(a) デマンド…親から子への要求
- プロセスの起動を要求するinvokeメッセージ

(b) イベント…子から親への応答
- 1個の解を返すsuccessメッセージ
- 解が無いことを知らせるfailメッセージ

しかし、実際にKPR上で論理プログラムを実行する場合、冗長なプロセスやプロセス間通信の存在はオーバヘッドとなるため、次のようなプロセスの割り付け戦略をとり、最適化を行っている。

i) 冗長なプロセスの除去…単一の子しかもたず、単にデマンドの通過点でしかないプロセスは生成しない。

ii) イベントの飛び越し…解を返すプロセスとその解を実際に必要としているプロセスとの間の通信に多くのプロセスが介在する場合は、中間のプロセスを飛び越して、直接、祖先のプロセスにイベントを返す。

### 2.2 ストリーム並列処理方式

KPRにおいては1個の節本体の処理が1個のSプロセスに対応するが、特に親プロセスからのinvokeメッセージあるいは子プロセスからのsuccessメッセージを受信してから、新しい子プロセスを起動するまでの処理を、Sプロセス中の更に細かいプロセスとみなし、これらを「サブプロセス」と名付けている。すなわち、ストリーム並列処理では、複数の枝分かれしたストリームが並列に処理される。この時、本体中の各ゴールについて、そのゴールの評価の直前までに生成された1本のストリームに対し、ゴールからの解を用いてストリームを枝分かれさせる。ここで、各ゴールの評価の直前におけるストリームの状態を表すものがサブプロセス情報である。すなわち、このサブプロセス情報は、並列実行下での各ゴールの環境である。

```
perm([], []).
perm([H|T], [A|P]) :- del([H|T], A, L), perm(L, P).
del([H|T], H, T).
del([H|T], L, [H|T2]) :- del(T, L, T2).
```

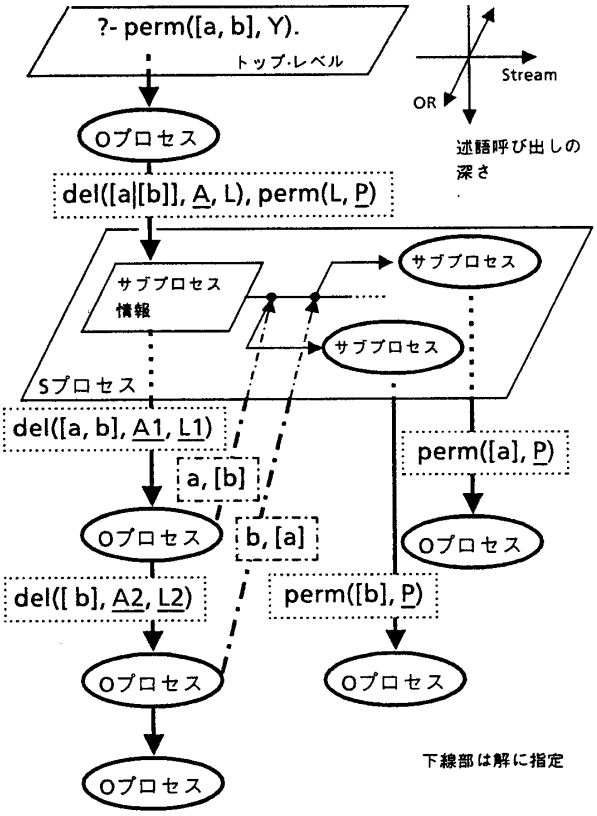


図1　論理プログラムの実行過程

図1に論理プログラムが実行される際に生成されるプロセスとその間のメッセージの例を示す。まず、Sプロセスは、invokeメッセージを受信すると生成される。次に、そのSプロセスは子プロセスのinvokeメッセージを作成し、送信するとイベント待ち状態となる。その後、起動している子プロセスからsuccessメッセージを受信するごとに、起動

*Stream Processing of a Logic Programming Language-Oriented Parallel Machine: KPR*
*Yasushi KAWAKURA, Kiyoshi SHIBAYAMA, Hiroshi HAGIWARA*
*Kyoto University*

直前のサブプロセス情報をコピーして解を取り込み、次のデマンドやイベントを送信する。

### 2.3 通信メッセージの構成

プロセス間で授受されるメッセージは、以下の情報で構成される。

(a) プロセス管理情報…プロセスの管理のための情報。
(b) 引数情報…述語呼び出しにおける引数を与える情報。
(c) 変数束縛情報…節内の変数がどのような値に束縛されているかを示す情報。
(d) 環境情報…構造データの実体、解指定情報。

また、解指定情報とは、解の送出先が必要とする変数を指定するための情報である。これは、KPRの実行モデルがイベント飛び越し戦略を採っているため必要となる。

OプロセスからSプロセスへのデマンドは、変数束縛情報と環境情報で構成され、SプロセスからOプロセスへのデマンドは、引数情報と環境情報とで構成される。

### 2.4 ゴールの実行環境

1個の節本体中に現れる変数は、節頭部で束縛された場合には、その節中では変化することはない。すなわち、Sプロセスにとっては静的(実行開始前)に決定していると見なされる。よって、変数束縛情報をすべてのサブプロセスで共有し、実行途中で束縛された値やヒープはサブプロセス情報としてコピーし、その中で保持する。

解指定情報では、デマンド送出時に未束縛の変数を指定する。これは実行時でなければ決まらない情報である。また、指定されるべき変数の位置はデマンド・メッセージ内とサブプロセス内とでは異なるので、それぞれのために解指定情報を作らなければならない。解情報は、この解指定情報によって指定された変数に対して束縛された値のみを返す。従って、これをサブプロセス情報に取り込む処理が必要になるが、プロセス間の通信量は削減できる。

## 3. ARUのハードウェア機構

### 3.1 KPRのシステム構成

KPRシステムは、並列リダクション・モデルの各種プロセスそれぞれの機能に対応した専用プロセッサから構成される機能分散処理システムである。論理プログラムの処理という観点からみると、Oプロセスを処理するORリダクション・プロセッサ(ORP)は単一化処理を、Sプロセスを処理するANDリダクション・プロセッサ(ARP)は環境の管理を行うプロセッサと言える。

ARPは、プロセスの管理をおこなうPCU-A (Process Control Unit) と本体ゴールの実行を行うARU (And Reduction Unit)とから構成される。図2に要素プロセッサの構成を示す。なお、図2中には、図1のSプロセスからデマンドを送出する直前の各情報も併せて示してある。

### 3.2 ARUのハードウェア構成

ARUでは、引数情報の作成、環境の更新、解情報の作成を主として行う。このために必要なハードウェア機構としては、以下のものがある。

●CM、LEMMとSPIM

他のプロセッサとの通信用にCM (Communication Memory)、LEMM (Local Environment Memory Module) があるが、サブプロセス情報を各ARUごとに格納するSPIM (SubProcess Information Memory)も持つ。

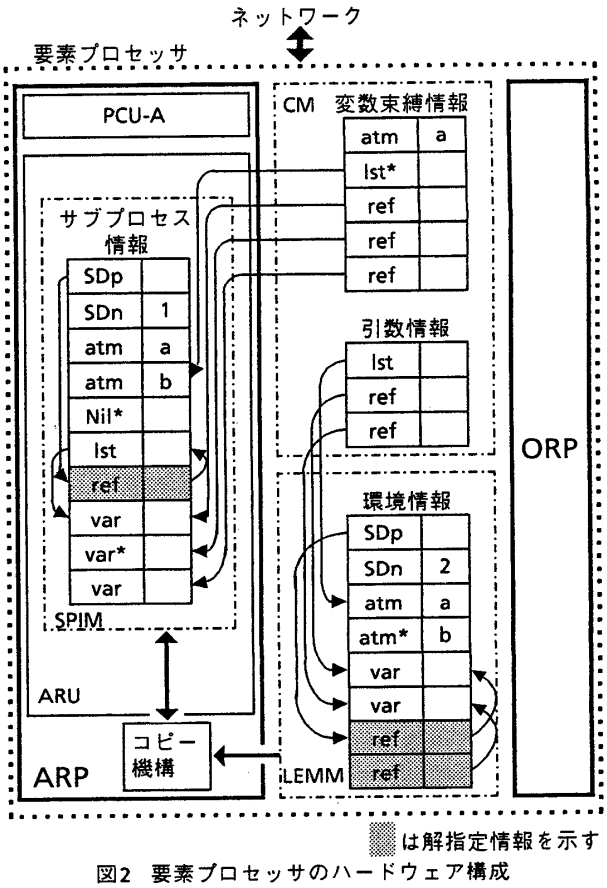


図2 要素プロセッサのハードウェア構成

●コピー機構

KPRでは、ストリームを枝分かれさせる時に、環境をコピーする方式を採用している。従って、環境のコピーによるオーバヘッドを少なくすることが重要になる。このため、専用ハードウェアによってARUでの処理と並行して、次に処理するゴールの環境をSPIM上に作る。

●間接参照の高速化

ARUにおいては、SPIMへの間接参照が多い。そこで、読み出したデータのタグによって、引き続き次のメモリ・アクセスを行えるようにアドレス・データのバイパスを設ける。

●イベント飛び越しのための機構

節の最右のゴールがユーザ定義述語の場合には、祖先に解を返さなければならないが、そのためにデマンドを出す時に親から受け継いだ解指定情報をそのまま指定する。この時は、そのゴールの処理開始前にあらかじめ解情報となる領域をLEMM上に確保しておく。

## 4. おわりに

KPRでは、環境コピー方式を採用したが、プロセス管理とANDリダクション処理との兼ね合いにより、必ずしもオーバヘッドが無くなるとは限らない。よって、プロセスの管理方式において詳細に検討しなければならない。

また、ストリーム並列処理方式は、1本のストリームに着目すれば逐次処理に過ぎず、AND並列型の言語を実行する際には、共有変数の無矛盾性を保証しなければならないという問題がある。