# ストリームに基づく並列オブジェクト指向言語 A'UM-90

## 2E−2 - ストリーム分散実装方式 -

丸山 勉、小西 弘一、小長谷 明彦、吉田 かおる†、近山 隆†

日本電気(株) C&C システム研究所 †(財) 新世代コンピュータ技術開発機構

### 1 はじめに

A'UM-90[1, 2] は、ストリームと呼ばれる非同期通信機構に基づく並列オブジェクト指向言語であり、第五世代計算機プロジェクトの一環として研究が進められている。これまでに逐次版処理系が完成しており、現在、分散処理系の開発を進めている。本稿では分散実装において最も重要となるストリームの実装方式について述べる。

ストリームの分散環境下での実装においては、効率的なメッセージ到着順の保証とPE間のメッセージ送信回数の最小化が、処理の高速化を考えた時に最も重要である。本稿では、この2点が2種類の制御メッセージにより効率的に実現できることを示す。

### 2 分散環境におけるストリーム計算の問題点

#### 2.1 ストリームの機能

A'UM-90 におけるストリームは、メッセージの通信路であって、メッセージの到着順序の保証、非同期通信、動的な生成、通信先の動的な決定等の特徴を持つ。ストリームには、頭部(inlet)と尾部(outlet)がある。ストリームに対するメッセージ送信は outlet に対して行なわれる。ストリームの inlet は、ストリームの outlet もしくはオブジェクトと接続することができ、ストリーム中のメッセージは、inlet 経由で接続先のストリームもしくはオブジェクトに転送される。

#### 2.2 逐次版処理系におけるストリームの実装

逐次版処理系では、ストリームはジョイントと呼ばれるデータ構造によって実現されている。inlet,outlet の区別はコンパイル時に行なわれており、処理系内部においては同一のジョイントを指す。ジョイントは無限長のキューとして実現されている。ジョイントの宛先が未確定の場合にはメッセージはこのキューに保持され、宛先が確定するのを待って次々と接続先のジョイントに転送される(実際にはポインタのデレファレンス(間接参照の短絡)が行なわれため、それ以降のメッセージは最終接続先のジョイントに直接送られる)。

#### 2.3 ストリームの分散実装の問題点

分散環境でのストリームの実装では、次のような問題点が生じる。異なるPE上にある複数のジョイントが次々と接続された場合には、接続を行なわれた回数分だけ、PE間でのメッセージの転送が行なわれてしまい(異なるPE間でのポインタのデレファレンスはメッセージの到着順等の保証を考えると非常に非効率である)、メッセージの転送回数を不必要に増大させる。また分散環境ではストリームに対する操作が、異なるPE間で全く非同期に行なわれるためメッセージの到着順を正しく保証するために何らかの手段が必要となる。

従って、分散環境下でのストリームの実装においては、以下の2点を実現することが重要である。

- PE間でのメッセージ送信回数の最小化
- (参照カウント操作メッセージを含む)効率的なメッセージ到着順の保証

### 3 ストリームの分散実装方式

上記の問題点を解決するために、PE間にまたがるストリーム操作に対しては、以下の基本手順をとることにする。

1. 接続先ストリームに WhereAreYou メッセージを送信
2. ストリームの接続先は未確定のまま(従って、それ以降このストリームに送信されたメッセージはこのストリーム内に保持され、接続先のストリームには送られない)
3. WhereAreYou メッセージに対する返答である IamHere メッセージの受信を待つ
   (WhereAreYou を受信したオブジェクトは自分自身の位置を引数として IamHere を返送する)
4. IamHere を受信したストリームは、接続先を IamHere の引数とする
5. 新たな接続先に対してメッセージ送信等を行なう

この手順の特徴は、一つ先のストリームのみに対して WhereAreYou を送信し、WhereAreYou がそれより先のストリームに転送されないことにある。一つ先のストリーム経由で WhereAreYou を送信することによって、一つ先のストリーム中のメッセージがオブジェクトに受理された後で WhereAreYou がオブジェクトに到着し IamHere が返送されるため、メッセージの正しい到着順を保証できると同時に、一つ先のストリームしか経由しないため、メッセージ転送回数を最小化することができる。

図1を例にとり、この手順を具体的に説明する。図1において、まずストリーム(B)とストリーム(C)が接続されたとする。この時、(C)中のメッセージを(B)に転送せずに、(B)に対して WhereAreYou を送信する。この WhereAreYou の引数は(C)の位置である。次いで、ストリーム(A)とストリーム(B)の接続が行なわれると、上記と同様にして WhereAreYou が(A)に対して送信される。最後に、オブジェクトとストリーム(A)の接続が行なわれると、(A)中のメッセージがオブジェクトに対して送信される。その結果、(B)から(A)に送られていた WhereAreYou がオブジェクトに到着し、その返答である IamHere(引数はオブジェクトの位置)が(B)に対して返信される。IamHere を受信することによって、(B)の宛先は、直接オブジェクトとなり(B)中のメッセージがオブジェクトに対して送信される。以下、同様にして(C)の宛先もオブジェクトとなり、(C)中のメッセージがオブジェクトに対して送信される。

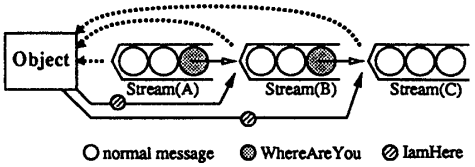


図 1: ストリームの接続の概要

上記の手順は基本的なものであり、それ以降そのストリームに対してメッセージが送信されないことが明確である場合、及びストリームが直接オブジェクトに接続された場合には、メッセージの直接転送等によるメッセージ送信回数の低

---

Stream Based Parallel Object Oriented Language A'UM-90
- Distributed Implementation of Stream -
Tsutomu Maruyama, Hirokazu Konishi, Akihiko Konagaya,
Kaoru Yoshida †, Takashi Chikayama †
NEC Corporation, †ICOT

減を行なうことが可能である。また、メッセージの複合化によるメッセージ送信回数の低減も可能である。

## 4 ストリームの分散実装の詳細

以下、ストリーム分散実装について詳しく述べる。

### 4.1 アドレス管理

あるPE内のオブジェクトのアドレスを他のPEに送信する場合に、分散環境では共有アドレスが存在しないため、何らかの変換が必要となる。本稿で述べる並列処理方式では、あるオブジェクトのアドレスは、一旦グローバルIDに変換され(アドレス輸出表に登録する)、他PEに送られる。また、メッセージの引数として輸入されたグローバルIDは、PE内のローカルアドレスに変換され用いられる(アドレス輸入表に登録される)。この結果、輸入表のエントリのみが他PE内の輸出表エントリへの参照となる。

### 4.2 ストリーム操作

以下ストリームの接続を例にとりどの様にWhereareYouメッセージとIamHereメッセージが用いられるか説明する。

図2にストリームの接続の様子を示す。図2(1)において、ジョイントPがPE#j上に、ジョイントQがPE#k上にあり、それぞれの宛先は未確定であるとする。また、Pのinlet、Qのoutletがそれぞれ PE#i に輸入され、PE#i上のオブジェクト(Obj#A)がそれ等を指しているとする。PE#i上のX,Yは、それぞれP,QがPE#iに輸入されたことによって作られたアドレス輸入表のエントリである。このエントリに要求される機能はジョイントと同様であることから、ジョイントとして実現される。

まず、図2(1)においてObj#Aが、ストリームPとQ(すなわちXとY)の接続を行なうものとする。Xはinletであり、Yはoutletである。従って、メッセージの流れる方向はXからYとなる。この時、直接XとYの接続を行なうのではなく、Qに対してWhereAreYouが送信される(図2(2))。この時、XとYがこの接続によってもはや参照されなくなった場合には削除される。このWhereAreYouは引数として、ジョイントPの位置を持つ。

次いで、PE#k上においてストリームQとPE#l上のオブジェクト(Obj#D)の接続が行なわれるものとする(実際にはジョイントQとZの接続が行なわれる)。この接続によって、ジョイントQに蓄えられていたメッセージはObj#Dに送られる。Obj#DはWhereAreYouを受け取るとその引数(この場合はジョイントP)に対してIamHereを送り返す。IamHereの引数はObj#Dの位置である。IamHereを受信することによってジョイントPのメッセージ送信先が初めて確定し、ジョイントPからオブジェクトObj#Dに対してメッセージが送られる。

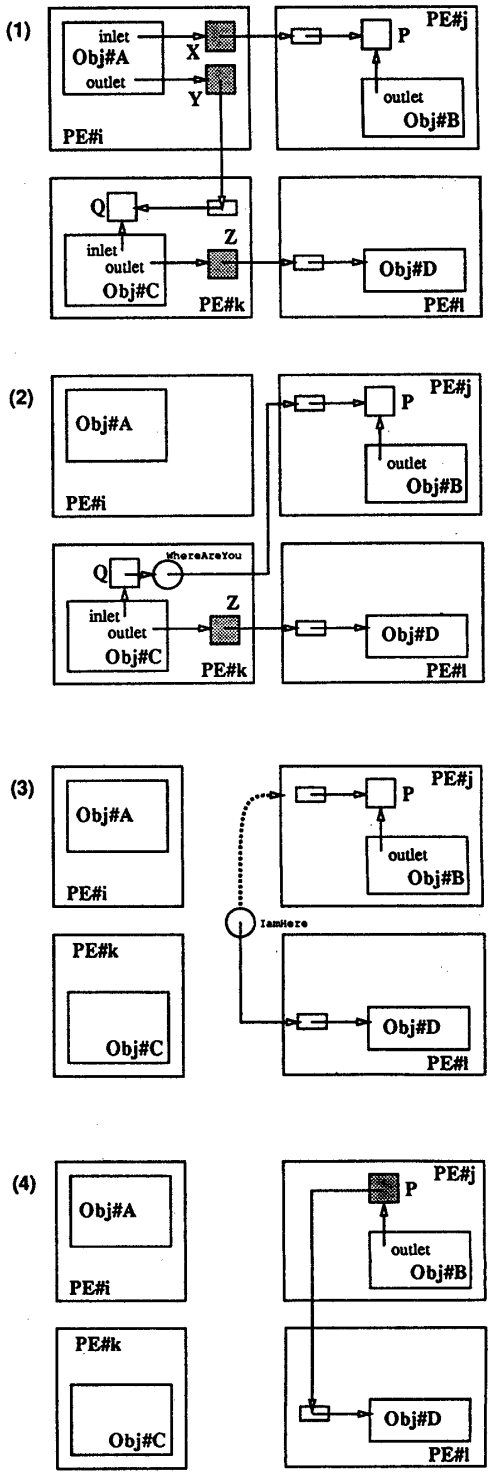


図 2: ストリームの接続

## 5 おわりに

以上、ストリームに基づく並列オブジェクト指向言語A'UM-90の実装において最も重要であるストリームの分散実装方式について述べた。本稿で述べた方式では、2種類のみの制御メッセージを用いることによって、メッセージ到着順の効率的な保証とPE間のメッセージ送信回数の最小化が実現できる。現在処理系を作成中であり、今年度中に完成する予定である。

## 謝辞

この研究の場を用意してくださったICOT内田研究部長、NEC C&Cシステム研究所の石黒所長、同研究所コンピュータ・システム研究部の小池部長、横田課長に感謝致します。

## 参考文献

[1] Kaoru Yoshida and Takashi Chikayama, "A'UM – A Stream-Based Concurrent Object-Oriented Language –", FGCS'88

[2] 小西, 丸山, 小長谷, 吉田, 近山, "並列オブジェクト指向言語A'UM-90", JSPP90