3E-8

ＯＺ：対象指向開放型分散システムアーキテクチャ

--- オブジェクトの表現形式と形式変換 ---

塚本享治（電総研）　四反田秀樹（松下電器）　舟渡信彦（シャープ）
吉江信夫（住友電工）　中込昌吾（ＡＢＣ）　田中伸明（松下電器）

１。まえがき

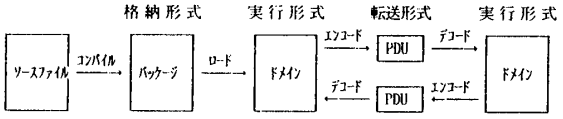
異機種計算機群上における分散型の高度な応用が簡単に行なえるようにすることを目指して、分散システム上でデータと手続きを一体化したオブジェクトを転送し操作することを特長とする対象指向型分散処理システムＯＺの開発を進めている[1-2]。本稿では、異機種で構成される分散システムであるが故に工夫したオブジェクトの表現方法と転送方法について述べる。

２。表現形式と形式変換の概要

対象指向型分散処理言語[2]で記述されたソースプログラムは、コンパイルし格納形式に変換して、パッケージと呼ぶファイルにタイプ単位で記憶する[2]。

ローダはパッケージからタイプを読み出し、ドメインと呼ぶ仮想機械のヒープにロードする。ドメイン内ではタイプとそのインスタンスはオブジェクトという単一の実行形式をとり、オブジェクト同士はドメイン内とドメイン間ではそれぞれポインタと識別子で関係づけられている。

オブジェクト間でオブジェクトを授受することで処理が進行する。異なるドメイン間でオブジェクト群[2]を転送するさいには、送信側で転送形式に変換（符号化）してパケットに詰め、受信側では逆変換（復号化）してオブジェクトに戻す。そのさい、転送されるオブジェクト群だけでなく、転送されないオブジェクト群も含めてシステム全体でオブジェクトの相互関係（トポロジ）を保存する。また、プロセスを転送するときには、移動先で再開できるように実行環境（コンテクスト）を修復する。



３。格納形式

プログラムはタイプごとにコンパイルする。異機種で実行可能にするため、Smalltalk-80流の仮想機械命令（バイトコード）を生成し、タイプ名をキーとしてパッケージと呼ぶISAMファイルに格納する。

タイプは継承関係を持つため、上位タイプを変更すると下位タイプの再コンパイルが必要になるし、下位タイプをロードすると上位タイプのロードも必要になる。このようなタイプの相互関係を管理するための辞書をパッケージ内に設けている。辞書の各エントリは、タイプ名、継承関係、タイプ作成時刻、下位タイプの再コンパイルが必要となった時刻、外部参照タイプ名、で構成される。

また、異機種間互換性を確保するために、符号化には ASN.1を用いている。

４。実行形式

ドメインのヒープ中ではすべてのものはオブジェクトの形態をとる。その表現形式は、メモリと処理の効率、ガーベッジコレクション、および転送時の形式変換の機械的処理、を考えて、細かく分類している。まず、１ワード(4バイト)で表現されるワードオブジェクトと連続した複数ワードで表現されるブロックオブジェクトとに分けられる。

ワードオブジェクトは下位２ビットのタグによってさらに、整数、実数、ポインタ(OP)、拡張(EXT) に分けられる。ここで、OPは以下に述べるオブジェクト表へのポインタであり、EXT はシステムが特殊用途に使用する。

ブロックオブジェクトは、検索、ドメイン間にまたがったOP、ドメイン間でのオブジェクトの移動、などを効率良く行なうために、エントリ部とセル部の２部分に分けている。次の構成をとるエントリ部が集まった辞書をオブジェクト表と呼ぶ。オブジェクトが他のドメインにある場合は、tagAndAtr が外部参照となり、 adrにdomainIdが入る。

```
typedef struct {
  long  tagAndAtr;      /*種類、GC制御ビット    */
  union {long *cell; long domainId;} adr;
  long  gid;            /*タイプ、シンボル、モニタインスタンス*/
  long  lid;} objEntry /*形式変換作業用      */
```

セル部はエントリ部の tagによって以下の種類に分けられ、ヒープにとられる。

(1) シンボル:　タイプ名、メソッド名、アクション 名 。

(2) タイプ:　パッケージ中のタイプをロードしたものであり、名前はOPに変換 。

(3) モニタインスタンス、クラスインスタンス、ロングアレイ、ショートアレイ、バイトアレイ:

OZ: Object Oriented Open Distributed System Architecture -- Object Syntaxes and their Transformations
Michiharu TSUKAMOTO[1], Hideki SHITANDA[2], Nobuhiko FUNATO[3], Nobuo YOSHIE[4], Shogo NAKAGOME[5], Nobuaki TANAKA[2]
1)Electrotechnical Laboratory　2)MATSUSHITA Electric Industrial Co.,Ltd.　3)SHARP Corporation
4)SUMITOMO Electric Industries, Ltd.　5)ABC Corporation,Ltd.

タイプのインスタンスであるが、それぞれに対応した効率良いメモリ表現をとる。

(4) プロセス、スタックブロック、キュー: モニタインスタンスを構成するもので、仮想機械が使用する。

セル部の主なものの形式を以下に示す。

```
typedef struct {
  objInt heapDead;
  objPnt type;
  ObjPnt superType;
  objInt typeAtr;
  long   instVQn;
  long   n_metInf;
  long   p_bytCod;
  long   p_litInf;
  long   p_insSymInf;
  long   p_linInf;
  long   p_tmpSymInf;
  metInf metInf[*];
  bytCod bytCod[*];
  litInf objPnt[*];
  insInf objPnt[*];
  linInf linInf[*];
  tmpInf tmpSymInf[*];
  } typeObj;
typedef struct {
  objInt heapHead;
  objPnt type;
  obj    instVar[*];
  } classInstObj
```

```
typedef struct {
  objInt heapHead;
  objPnt type;
  objInt miAtr;
  objInt msgCnt;
  objPnt entq;
  objPnt recq;
  objPnt rstq;
  obj    instVar[*]
  } monitorInstObj
typedef struct {
  objInt heapHead;
  objPnt type;
  objPnt prev;
  objPnt next;
  objPnt monInst;
  objExt errObj;
  byte   *pc;
  long   *fp;
  long   *sp;
  long   *ap;
  objPnt stack;
  objInt msgId;
  objInt waitMsg;
  long   trapWord;
  long   rstCode;
  } processObj
```

5。転送形式と符号化・復号化

実行形式のブロックオブジェクトはOPによって連結されており、プロセスやスタックは実行時のコンテクストを記憶している。これらを転送するさいに、オブジェクト群のトポロジを保存し、モニタインスタンスを移動先で実行再開可能とするため、以下のような転送形式をとっている。枠組は ASN.1に沿うが、実行形式中に含まれるオブジェクトの種類と長さの情報を生かして、オブジェクトの符号化はＯＺに固有なものとなっている。 ASN.1符号化規則に比べると転送形式の長さは⅔程度になる。

```
ozObjCmd ::= [APPLICATION n] CHOICE {
  call       [0] IMPLICIT requestObj;
  send       [1] IMPLICIT requestObj;
  return     [2] IMPLICIT resultObj;
        . . . . . .                  }
requestObj ::= SEQUENCE {
  cmd        [0] IMPLICIT comForm;
  entry      [1] IMPLICIT obj;
  args       [2] IMPLICIT SEQUENCE OF obj;
  objects    [3] IMPLICIT blockObjStream; }
        . . . . . .
obj ::= EXTERNAL
blockObjStream ::= SEQUENCE {
  objTable   [0] EXTERNAL;
  objCells   [1] EXTERNAL; }
```

ここで、EXTERNAL部分がＯＺ固有な部分である。

各コマンドによって決まる範囲内のオブジェクト[2]に対して次のような符号化と復号化を行なう。ただし、説明の簡単化のために、符号化と復号化は並行処理しないものとし、ＧＣに対する措置も省略する。

[符号化]

(1) blockObjStreamはドメインの構成と同様に、オブジェクト表とセル群から構成し、それぞれのエントリとセルは 連続して詰める。

(2) 範囲内のブロックオブジェクトの集合をS_1, S_1のエントリとセルの集合をそれぞれ、S_1eとS_1cとし、S_1cが直接指す 範囲外のエントリの集合をS_2*とする。S_1eとS_2eのlidにはblockObjStream内で一意な番号を入れ、(3)と(4)を行なったのち、クリアする。

(3) objCellsにはS_1cを詰める。ただし、各セルの 先頭にはlidと長さを詰める。 OPならそれが指す エントリ部のlidを OP化し、それ以外ならばそのまま 詰める。 また、プロセスとスタックの場合には、戻り番地とスタックの位置をそれぞれ、タイプ内および スタック内オフセットに変更して詰める。

(4) objTableにはS_1eとS_2eを詰める。ただし、objTable内のadrは、S_2eに属するモニタインスタンスの場合には送信側domainId、S_2eに属するタイプとシンボルの 場合はクリア、外部参照の場合は 参照先のdomainIdとする。また、S_1eに属するモニタインスタンスの場合には、オブジェクト表エントリ部のadrに 受信側domainIdを 入れて、セル部およびそれが指すスタック、プロセス、クラスインスタンスの エントリ部と セル部を解放し、クラスインスタンスの場合は、エントリ部と セル部を解放する。

[復号化]

(1) objTableからエントリ部を取出す。モニタインスタンス、タイプ、シンボルの場合、gidが一致するものがなければ オブジェクト表に登録し、一致するものがあればオブジェクト表のエントリを取出したもので 上書きする。外部参照の 場合、gidが一致するものがなければ オブジェクト表に登録し、一致するものがあればlidだけを上書きする。それ以外の場合は、無条件にオブジェクト 表に登録する。

(2) objCellsの セル部に必要な領域を獲得し、オブジェクト表エントリ部のadrに 登録する。この領域にobjCellsから 取出してコピーする。ただし、OPならその値と一致するlidを持つ エントリ部を探してそのエントリのアドレスをOP化し、それ以外ならばそのままコピーする。

(3) blockObjStreamに 含まれていたモニタインスタンスのプロセスとスタック中の 戻り 番地とスタックの位置を 実行可能なように修復する。これは(2)の 終了前には行なえない。

(4) オブジェクト表エントリ部のlidをクリアする。

6。あとがき

コンパイル、ロード、および転送は、すべて形式変換と考えることができる。ＯＺでは、パッケージの格納形式、ドメインの実行形式、パケットの転送形式、のいずれにおいても、オブジェクト間の複雑な関係が表現できる。関係の表現は、格納形式では名前、実行形式ではオブジェクトポインタとオブジェクト識別子、そして転送形式ではオブジェクト識別子で行なわれるが、これを容易にしているのはオブジェクトの登録辞書である。

現在、モニタインスタンスの移動を除いて実現されており、本方法が有効なことが確認されている。

参考文献
主題はすべて、『OZ: 対象指向開放型分散システムアーキテクチャ』
[1] 塚本、棟上: 情処、マルチメディア通信と 分散処理研資 34-7(62,7)
[2] 塚本他：情処36回全国大会4G-1~5 (63,3)