オブジェクト指向分散環境 OZ++の異機種間オブジェクト転送

7L－6

濱崎 陽一 (電子技術総合研究所)　中村 章人 (電子技術総合研究所)
鈴木 敬行* (シャープビジネスコンピュータ)　塚本 享治 (電子技術総合研究所)
*: 開放型基盤ソフトウェアつくば研究室研究員

1 はじめに

OZ++ は、クラスおよびオブジェクトをネットワーク上で共有、転送できるオブジェクト指向分散環境である。クラスの実行コードおよびオブジェクトの構造は機種に依存するために、異機種間で転送する際には工夫が必要である。クラスの実行コードについては、機種ごとにコードを用意して、機種を指定してロードすることにより解決する。ここでは、オブジェクトを異機種間で転送する機構について述べる。

2 レイアウト情報

ネットワークでオブジェクトのような複雑な構造を持つデータを転送する場合、ASN.1 などを用いてタグ付きの標準形式にエンコードする方式と、メモリイメージに近い形でエンコードする方式があり、変換が高速に行なえるために OZ++ では後者を用いている。

しかし、異機種間でオブジェクトの交換を行なうには機種の組み合わせの種類の変換に対応せねばならなくなる。この問題を解決するためにエンコードされたものは標準となる機種のメモリイメージに準拠したものにする事とした。また、どの機種にエンコードされたかはパケットに含まれるアーキテクチャ情報により示す。アーキテクチャ情報には、エンディアンなどアーキテクチャ固有の特性の情報が含まれる。

この方式では、特定の機種でのメモリレイアウトを標準機種のそれに変換するための情報が機種の数だけあれば足りる。この情報をレイアウト情報と呼ぶ。レイアウト情報は機種毎のクラスの実行コードを生成する際に、コンパイラにより生成され、クラス管理系から取得することができる。レイアウト情報は、標準機種と特定機種

```
Class B : A {
    long    X;
    int     Y;
    C       Z;
}
```

図 1: クラス B のプログラム (部分)

でのメモリレイアウトについてインスタンス変数の対応関係とメモリ上の出現順序の情報を持っている。

例として図1にプログラム (インスタンス変数の定義部分) を示したクラス B を考える。クラス B はクラス A を継承しており、プライベートなインスタンス変数 X,Y,Z を持っている。Z はクラス C のインスタンスへのポインタとして実現される。このクラスのアーキテクチャ α のレイアウト情報と、レイアウト情報によって標準機種のメモリレイアウトに変換する例を図 2に示す。

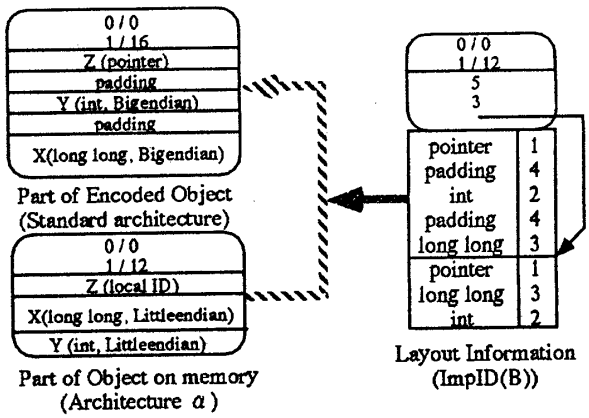

図 2: レイアウト情報とレイアウトの変換

3 オブジェクトの実装とエンコード

OZ++ のオブジェクトは、インスタンスを構成する各クラスの部分 (パート) を集めた構造になっている。クラス B のオブジェクトは、ヘッダとクラス A のパートおよびクラス B のパートから成る。これは、OZ++ ではクラスはバージョン管理されており、オブジェクトを構成するクラスの実装バージョンはコンパイル時ではなくインスタン生成時に決定されるためである。祖先クラスも含めた実装バージョンの組に対してコンフィギュレーションクラス ID(CCID) が付与され、オブジェク

An Implementation of Object transfer between different architecure machines in OZ++: An Object-Oriented Distributed Systems Environment
Yoichi Hamazaki(Electrotechnical Laboratory),
Akihito Nakamura(Electrotechnical Laboratory),
Takayuki Suzuki*(Sharp Business Computer Software, Co., Ltd.),
and Michiharu Tsukamoto(Electrotechnical Laboratory)
*: Researcher, Tsukuba Laboratory, Open Fundamental Software Technology Project

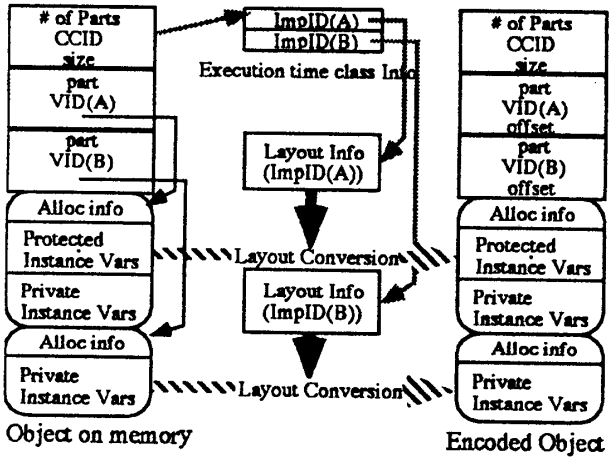

図 3: オブジェクトのエンコード

トはヘッダにその情報を持つ [1]。オブジェクトのエンコードの際には、各クラスのパートがそれぞれのクラスのレイアウト情報に従って変換される。各クラスの実装バージョンは、CCID をキーに実行時クラス情報を求め、そこから得られる。ポインタはパケット内でのみ有効なローカル ID やオフセット値に変換され、再帰的にエンコードされる [2]。デコードは逆の操作となる。

4 エンコード方式の検討

標準機種を定める事で機種の数だけレイアウト情報を用意すればあらゆる組み合わせの異機種間でオブジェクトの交換が出来る。一般に一つの組織内で使われる計算機は同じ機種のものである場合が多い。そうした同機種から成る環境において、それが標準機種でないからといってエンコード・デコードの際に標準形式に変換するのは妥当ではない。そこで対案として、送信時にはレイアウトの変換を行なわずに受信時にのみレイアウトの変換を行なう方式を考える。この場合には、受信側で送信側と受信側の機種のレイアウト情報を併せて用いることによりレイアウトの変換を行なう。前者を標準形式型、後者をソース形式型のエンコードと呼ぶ (図 4)。

この方式の利点は、1) 同機種計算機間ではレイアウト変換が起らない、2) 異機種間でもレイアウト変換が受信側でしか起らないので高速になる事である。また、欠点として受信側で他の機種のレイアウト情報を管理する必要が有る。レイアウト情報は実行コードと同時にロードするなど、その機種の情報のロードについては大きなオーバヘッドにはならないが、他の機種のレイアウト情報はロードのコストが大きい可能性が有る。

以上より、同機種計算機が多い場合には有効な方式といえるが、多様な機種間でオブジェクトを交換する際には適さない。よって、機種の多様性は組織毎に異なるために、どちらの方式をとるかは組織毎 (サイト毎) に選択する事とした。

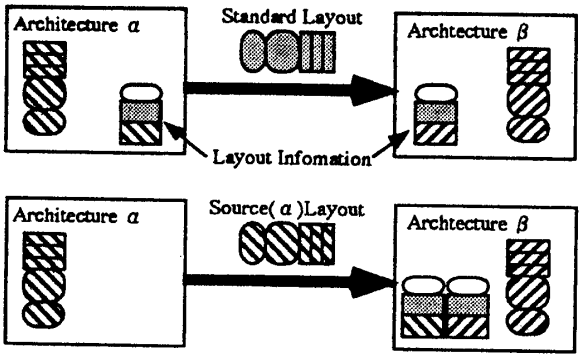

図 4: 標準形式型とソース形式型のエンコード

更に、標準形式に加えて、サイト毎の標準形式を導入すると、レイアウト情報をロードする可能性は低減できるが、管理すべきレイアウト情報が増えるため、大きな効果は期待できない。機種について事前に交渉することは可能であるが、異機種である場合にはレイアウト変換は必須であり、またレイアウト情報の有無を通信のたびに交渉するのはコストが大きすぎる。

サイト間の通信については、他のサイトについての情報を得るコストが大きく、アプリケーションゲートウェイでの中継が複雑になるために、標準形式型のエンコードを用いるのが適していると考えられる。

5 まとめ

OZ++ において異機種計算機間でオブジェクトを転送する機構について述べた。標準機種を定め、コンパイラにより生成されるレイアウト情報を用いてレイアウトの変換を行なう。標準形式型とソース形式型のエンコードする方式について得失と適用を検討した。

OZ++ システムは現在までに、SPARCstation 上で動作する第一版を開発し、インタネット上で公開している。今後、IBM-PC をターゲットに移植を行なう予定であり、ここで述べた方式で実装・評価を行なう。

本研究は、情報処理振興事業協会 (IPA) の「開放型基盤ソフトウェア研究開発評価事業」の一環として行われたものである。

参考文献

[1] 西岡、濱崎他:" オブジェクト指向分散環境 OZ++ システム第一版の実現 "、SWoPP'95, プログラミング研究会、Aug. 1995.

[2] Hamazaki et.al: "The Object Communication Mechanisms of OZ++: An Object-Oriented Distributed Programming Environment", IEEE 9th Int. Conf. on Information Networking(ICOIN-9), Dec. 1994.