3R－4

分散オブジェクトの開発支援について

井川 大介　江口 敦子　落合 正雄
(株) 東芝 府中工場

1.はじめに

オブジェクト指向技術をネットワーク上に展開するにあたって、オブジェクト間の通信を取り持つ機構を一般に Object Request Broker (ORB) と呼ぶ。現在、OMG(Object Management Group) においてORBの標準化 (CORBA; Common ORB Architecture) が進行中であるが、市販のORB製品においては、CORBA標準が確定していない部分での製品間の違い (特にC++ 言語での記述部分) が大きい。

本稿では、2つのORB製品 (Solaris* NEO、Orbix) を使用して分散オブジェクトシステムの開発を行った経験をもとに、その開発支援のソフトウェア・アーキテクチャについて考察する。

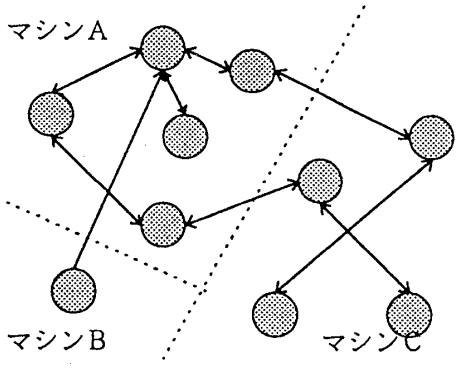

図1 分散オブジェクト

2.分散オブジェクト

ネットワーク上のオブジェクトを特に分散オブジェクトと呼ぶ。一般に、ORB製品を利用して分散オブジェクトを作成するには、以下の手順を用いる。

1. (Interface Definition Language) により、オブジェクトのインターフェースを記述する。
2. IDL コンパイラにかけ、「サーバサイド」と「クライアントサイド」の骨組みコードを生成する。
3. サーバサイドの骨組みコードに、オブジェクトの実体を実装する。
4. 目的言語コンパイラを起動し、実行形式とする。

ここで、サーバサイドとは「オブジェクトの動作を提供する側」のことであり、「クライアントサイド」とは、「オブジェクトを使う側」のことである。IDL コンパイラは、サーバとクライアントの間の通信を行うコードだけを生成するので、オブジェクトの実体や使うためのコードは自力で記述する必要がある。

3.開発上の問題点

前節で記述した手順により分散オブジェクトを生成することができるが、通常のC++ での開発に比べ、以下のような問題がある。

3.1.IDL から C++ へのマッピング

IDL コンパイラは、IDL で書かれたクラス宣言をC++のクラス定義に変換するコンパイラである。IDLはC++に似ているが異なるものであるので、「IDL からC++へのマッピング」が定義される。

IDL から C++へのマッピングにおいては、引数がメソッド「に」渡されるのか、メソッド「から」返されるのかで異なった C++マッピングを用いる。また、CORBA 1.2ではC++マッピングが未定義であったために、CORBA 1.2準拠であるSolaris NEO、Orbix、およびCORBA 2.0 それぞれでC++マッピングが少しずつ異なる。

3.2.メモリ管理

オブジェクトの実体を C++で実装する際に、IDL コンパイラが生成したコードがどのようにメモリを使用

Experience Report on Distributed Object Environment
Daisuke IKAWA, Atsuko EGUCHI, Masao OCHIAI
TOSHIBA CORPORATION

するか検討する必要がある。たとえば、Solaris NEO においては、メソッドを呼び出すと引数に使われていたメモリが解放されるため、あらかじめ「引数用に」メモリを確保する必要がある。また、IDL で配列を記述すると、特別なメソッド群を用いて「配列の実体」を確保しなくてはならない。

3.3.オブジェクト・サービスの実装

オブジェクト・サービスとは、オブジェクトの生成から破壊までに関わる「ライフサイクル・サービス」、オブジェクトの名前を管理する「ネーム・サービス」などがあるが、Orbix ではほとんど実装されておらず、自前でなんでも実施する必要がある。

3.4.デバッグ

近年は C++のソースデバッガが市販されるようになってきており、オブジェクトの動作を追跡できるが、CORBA 環境において異機種にまたがるオブジェクトの動作を追跡できるデバッグツールが必要である。

4.ソフトウェア・アーキテクチャ

前節であげた問題点を解決するために、筆者らは以下のソフトウェアを供給する必要があると考えている。

4.1.プリコンパイラ

CORBA 環境下では、まず IDL でインターフェースを定義し、実体を C++で書くという二重構造をもつために、言語のマッピングの問題など複雑な問題が発生する。しかし、従来からある C/C++のコードを利用するには、オブジェクトの実体を C++で実装することが必要である。

ここで C++を目的言語とするようなプリコンパイラを用いて、IDL 言語コード、ならびに IDL コンパイラが生成する C++コードに合う C++コードを生成することとすると(図 1)、

- IDL 言語の知識がいらなくなる
- ORB 製品によって異なる部分を吸収する
- メモリ管理の一部を自動化する
- C++で書かれた従来資産を生かす

といったことができるようになる。

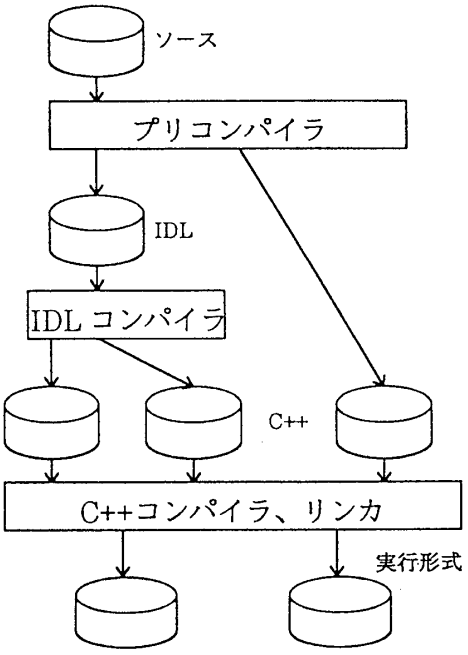

図2 プリコンパイラの概念

4.2.デバッグ機能

プリコンパイラを使うことで、C++コードのメソッド呼び出しや関数の出入り口などにコードを埋め込むことができるようになる。これを利用して、

- デバッグ情報の出力
- プロファイル情報の出力
- ORB が発生する例外のキャッチ

を行わせることができる。この出力を収集し、適当な形で見せることで、オブジェクト間の接続のデバッグ、オブジェクト配置の最適化が行えるようになる。

5.おわりに

分散環境下でオブジェクト指向技術を用いる開発が現実のものとなるには、標準化や実装技術の一層の進歩が必要であると考えられるが、普及が進んだ言語である C++を実装言語として使う例が増えてくると考えている。そのときに環境が要求する難しい問題を言語レベルで吸収すれば、実装の平易化が実現できるものと考える。

6.参考文献

OMG: CORBA 2.0 Specification

(*) Solaris は、米国における米国 Sun Microsystems, Inc. の登録商標です。