3R－6

分散オブジェクトプラットフォームの開発(2)
-プログラミングモデル-

乙川 進一　森 健
沖電気工業株式会社 マルチメディア研究所

1 はじめに

ネットワークの普及に伴い，アプリケーションプログラム (AP) の分散環境への対応が求められている．APを分散環境中で動作させる典型的な方法としては次の3つがある．1. 既存のプログラミング言語からOSのシステムコールやライブラリを呼び出す方法．2. 既存のプログラミング言語とRPC用スタブジェネレータを組み合わせて用いる方法[1]．3. 分散プログラミングを可能とするように設計された新たな言語を用いる方法[2]．第1，第2の方法は，多くの開発環境で利用可能であり，広く使われている．しかし第1の方法には，その低水準なインタフェースによって，プログラムの生産性や可読性が低くなるという問題がある．第2の方法は，より高水準なプログラミングを可能としているが，通信相手の決定方法や交信形式が強く固定されており，あらかじめ想定されている形態と異なる通信の実現は必ずしも容易ではない．一方，第3の方法は，プログラムの移植性，有用なライブラリ資産の利用可能性などの問題が解決される限りにおいては，理想的な解決を提供し得るが，新しい言語という点に問題がある．

そこで，我々はC++言語の特性である「クラス」という型に注目し，第1 / 第2の方法によって，第3の方法を近似的に実現する分散オブジェクトプラットフォーム（以下分散プラットフォームと呼ぶ）の提案を行なっている．本稿では現在開発中の分散プラットフォームに関して，APのプログラミングについて述べる．

2 分散プラットフォーム

我々が開発中の分散プラットフォームでは，2つの分散オブジェクトが存在する．1つはサーバオブジェクト，もう1つはクライアントオブジェクトである．サーバオブジェクトはさらに，揮発オブジェクト，永続オブジェクトに分けられる．永続オブジェクトは分散プラットフォームが停止した後でも，オブジェクト内部の情報を2次記憶上に保持し続けるが，揮発オブジェクトはそのすべて情報を失なう．

図1は分散プラットフォームの概観である．分散オブジェクトを1つの実行主体として実現するために，分散オブジェクトとメッセージの送受信を管理する（次節で説明する）スタブを一体化したプロセスとして表現している．

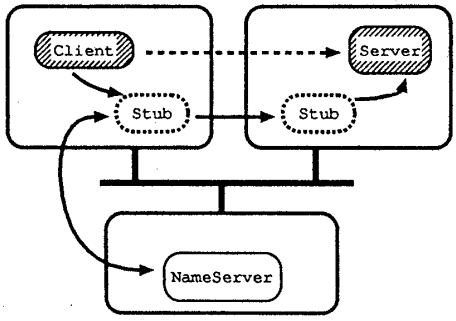

図 1: Distributed Objects Platform

3 分散オブジェクトの作成

この節では，スタブジェネレータを用いた，分散オブジェクトの作成について述べる．分散オブジェクトの作成はC++の定義によって行う．

3.1 スタブの生成

分散オブジェクトはC++ライクな定義を用いて，インターフェースとインプリメンテーションを記述する．C++で記述した分散オブジェクトを分散プラットフォーム上で動作させるために，スタブジェネレータによってスタブを生成する．生成されるスタブは**Server Stub**と**Client Stub**があり，C++のクラスとして実現されている．スタブでは，プロセス間通信のためのコードを埋め込まれた分散オブジェクトのメンバ関数（メソッド）が用意されているので，メンバ関数を呼べば，プロセス間通信を全く意識することなく，分散オブジェクトを開発／利用できる．図2はスタブジェネレータの概観である．

A Distributed Objects Platform(2) – Programming model–
Shin-ichi OTOKAWA, Takeshi MORI
Oki Electric Industry Co.,Ltd.
550-5 Higashiasakawa, Hachioji, Tokyo 193, Japan

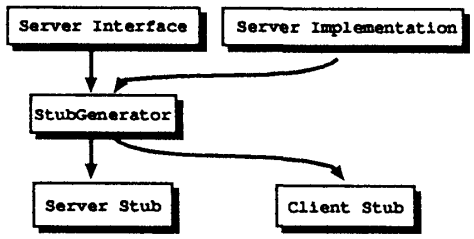

図 2: Stub Generator

3.2 通信のためのインタフェース

スタブジェネレータによって生成されたスタブ自身は，通信のためのインターフェースを持っていない．通信のためのインターフェースはスタブが継承するクラスで提供される．スタブが継承するクラスを変えることによって，様々な通信方法に対応できるようになる．通信のためのインターフェースは次のものである．

initiate(Server stub) ネームサーバ （[3]を参照）にオブジェクトの情報を登録し，クライアントからメッセージを受け取るための初期化を行う．

terminate(Server stub) オブジェクトを停止する．

activate(Server stub) サーバがクライアントから，どのメソッドを起動するかを表すデータを受取り，そのメソッドの処理を行う．

connect(Client stub) クライアントがサーバに接続する．

send,receive(Server/Client stub) それぞれデータを送信，受信する．

このように，実際の通信の実現部と，スタブを分離することによって，どんな通信方法を使おうとも無関係にスタブを生成できる．

3.3 オブジェクトの作成

分散オブジェクトを作成／利用するためには，スタブで提供されるメンバ関数を明示的に呼出せばよい．ただし，永続オブジェクトを作成する場合には，次のメンバ関数を記述しておかなければならない．

coders 保持しておきたいオブジェクトの内部の情報を符号化する．

decoders 保持しておいた情報を復号する．

saveState coders で符号化したデータを 2 次記憶に記録するための処理を記述する．

restoreState 2 次記憶に記録したデータを decoders で復号するための処理を記述する．

これらはスタブが継承している d_Coder, d_Decoder, Persistence クラス（図 3を参照）のメンバ関数である．これらはすべて仮想関数として実現されている．

4 オブジェクトライブラリ

スタブジェネレータから生成されたスタブが利用するクラスは図 3 のような関係にある．**ServerRoot**,

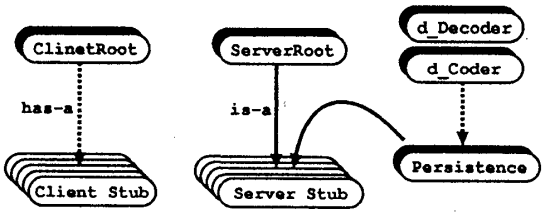

図 3: libraries

ClientRoot はそれぞれ，サーバ，クライアントオブジェクトが必ず利用するクラスであり，実際の通信の詳細部分を実現している．様々な通信方法について，このクラスを用意しておけば，分散オブジェクトの記述を変更することなく，どんな通信方法にも対応できる．**Persistence** は永続性を実現するためのクラスである．**d_Coders, d_Decoders** はそれぞれ，基本データ型をバイト列，バイト列を基本データ型をに変換するクラスである．

5 まとめ

本稿では分散オブジェクトプラットフォームでのプログラミングについて説明した．本分散プラットフォームを利用すれば，通信方法に依存しない分散アプリケーションの構築が可能となる．今後は，オブジェクトのセキュリティなどの機能を取り入れて行きたい．

参考文献

[1] S.K. Shrivastava, G.N. Dixon, and G.D. Parrington, "An Overview of the Arjuna Distributed Programming System", IEEE Software, No.1, 1991

[2] A.Yonezawa, "ABCL : An Object-Oriented Concurrent System", The MIT Press, 1990

[3] 森他, "分散オブジェクトプラットフォームの開発(1) – 概要–", 情報処理学会第 52 回全国大会予稿集 3R–5, 1996