分散オブジェクトプラットフォームの開発(1)

3R－5

－概要－

森健　乙川進一

沖電気工業株式会社 マルチメディア研究所

1　はじめに

オブジェクト指向による分析・方法論の進化や計算機環境の分散化に伴い、ミドルウェアとして分散オブジェクト環境が注目されている。分散オブジェクト環境の標準的仕様としてはOMG(Object Management Group)のCORBA(Common Object Request Broker Architecture)[1]があるが、CORBAベースの環境は汎用性を重視しているため、特定分野の専用のシステムに適用するには使いづらい面もある。

我々は分散アプリケーション(AP)の作成を支援するための、分散オブジェクトプラットフォームを開発中である。CORBA準拠の環境よりも自然で容易な分散オブジェクトの作成／利用を狙ったものであり、環境自体もより小型なものとしている。本稿では本プラットフォームの概要について述べる。

2　設計方針

CORBA環境では、以下のような問題点がある[2]。

・多言語環境のためオブジェクトの記述が複雑。
・APからのオブジェクトのオペレーションの呼び出しが自然に記述できない。
・環境にODB等を備えるため、多大な資源が必要。
・環境内部での処理が見えず(内部の実装は自由なため)、手も入れられないため、ORB内でエラーが起きたり処理速度に問題があっても修正できない。

これらの解決と、CORBAでの利点(オブジェクトの位置透過性等)を備えた環境を目指した。また、高信頼性／対障害耐久性を要求されるシステムにも適用できることを目標とした。同様な環境としてはArjuna[3]等があるが、将来的に記述力向上のために言語拡張する可能性を考え、既存システムの利用は避けた。以下に主な設計方針を示す。

1.C++ベースの単一言語環境とする。
2.スタブジェネレータにより、C++のオブジェクト定義からローカル(AP)側のスタブオブジェクトとリモート側のサーバオブジェクトのスケルトンを生成する。
3.サーバオブジェクトの位置はネームサーバにより管理する。
4.オブジェクトを不揮発性記憶に退避するためのオブジェクト符合化／復号化機能と、これを用いた永続オブジェクト及びアトミック操作を持ったオブジェクトの作成用ライブラリを備える。

1、2によって、AP及び分散オブジェクトの記述を大幅に簡易化している。一つのオブジェクトは実体であるサーバオブジェクトとアクセス用のスタブとから構成され、サーバオブジェクトは唯一の存在だが、それに対するスタブは複数存在してよい。この時、アクセス速度の向上のためにスタブにデータを置くことも考えられるが、データの一貫性制御が必要になることを避け、現在は単なるアクセスの口としている。

また3によってサーバオブジェクトの位置がスタブ生成時に静的に固定されず、動的に決められる。これによりシステム構築の柔軟性が高まる。この結果、サーバオブジェクトを探すために必ずネームサーバへのアクセスが必要となるが、この処理をスタブ生成処理の内部に隠蔽した。これにより、APの記述の際にはネームサーバへのアクセスを全く意識せずにすみ、APの記述がより自然で簡潔になった。また、ネームサーバへのアクセスをこのスタブ生成時の一回のみとし、以後のサーバオブジェクトへの操作依頼は直接スタブーサーバオブジェクト間で通信

A Distributed Objects Platform (1) Overview
Takeshi Mori, Shinichi Otokawa
Oki Electric Industry Co., Ltd.
550-5 Higashiasakawa,Hachiouji,Tokyo 193,Japan

し、ネームサーバを介さないようにすることで、速度上のオーバヘッドを抑えた。

更に4によりオブジェクトの不揮発化の枠組を提供し、そこからのデータ復旧を行うことにより信頼性／対障害性の向上を実現している。

3 プラットフォーム概要

本プラットフォームの構成要素を以下に示す。

- オブジェクトライブラリ：通信、オブジェクトの符合化／逆符合化、永続化、操作のアトミック化のための C++オブジェクトライブラリ。
- スタブジェネレータ：C++のオブジェクト定義から、スタブ&サーバオブジェクトのスケルトンを生成する。
- 各種サーバプロセス：プラットフォーム管理用のマネージャ／サブマネージャ、オブジェクトの位置管理をするネームサーバ、各分散(永続)オブジェクトのサーバ。

オブジェクトライブラリとスタブジェネレータ[4]は、オブジェクトと AP の作成時に使用する。ユーザはまず C++によって目的のオブジェクトを定義する。ここで、永続オブジェクトやアトミック化された操作を持ったオブジェクトを作りたい場合には、オブジェクトライブラリで提供されているクラスを用いる。次にそれをスタブジェネレータにかけて、当該オブジェクトのスタブ&サーバのスケルトンのソースを生成する。これにはオブジェクトライブラリで提供されている通信用クラスが継承される。そして生成されたソースを元に、操作の具体的内容等を埋め込んで当該オブジェクトを完成させる。

一方、AP からのオブジェクトの利用は、ここで生成したスタブを用いて記述する。スタブ自身も C++オブジェクトであるため、記述は全て通常の C++プログラミングの要領で行うことができる。

また、各種のプロセスは AP の実行を支援するものである。管理マネージャ／サブマネージャはプラットフォームの起動／終了、動作状況管理等を行うためのプロセスであり、マネージャはプラットフォーム全体に唯一、サブマネージャはプラットフォームを構成する各マシン毎に一つ存在する。また、ネームサーバはオブジェクトの名前とその位置を管理するプロセスであり、プラットフォーム全体に一つ存在する。更にサーバオブジェクトは各オブジェクトの実体であり、プラットフォーム起動時にはサブマネージャによって、ユーザに指定されたオブジェクトや永続オブジェクトのサーバオブジェクトが起動される。それ以外のサーバオブジェクトは、AP によってそのオブジェクトに対する最初のスタブが生成された時に起動される。この処理はスタブの生成関数の内部で行われる。具体的な処理としては、生成関数はまず当該オブジェクトが既に存在するか否かをネームサーバに問い合わせ、存在しない場合には指定されたマシン上のサブマネージャにサーバオブジェクトの起動を依頼する。

図1に、本プラットフォームの稼働時の構成を示す。

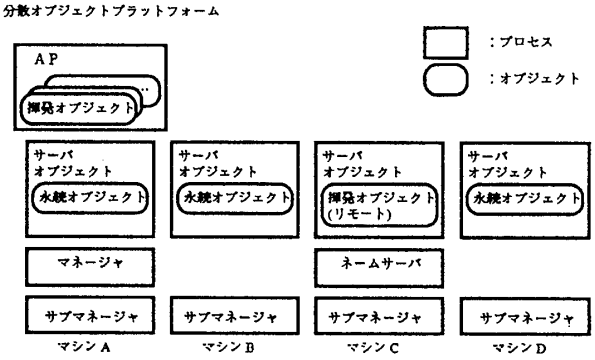

図 1: 分散オブジェクトプラットフォームの構成

4 おわりに

分散AP の開発を支援する分散オブジェクトプラットフォームの概要について述べた。本システムはワークステーション上で α 版が稼働中である。今後は処理速度や実用面での評価／改良と、PC 系 OS への移植を予定している。

参考文献

[1] 大野他, "オブジェクト・マネージメント・グループとその活動", 情報処理学会誌, Vol.35, No.9, pp.845-852, 1994

[2] 乙川他, "分散ソフトウェア開発用ツールキット", 情報処理学会第 48 回全国大会予稿集 6D-8, 1994

[3] S.K.Shrivastava 他, "An overview of the Arjuna distributed programming system", IEEE Software, January 1991

[4] 乙川他, "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", 本大会予稿集, 1996