2N－5

CORBA準拠ORBのリアルタイム化における アプリケーション記述支援機構

中村 隆幸　武本 充治　田中 聡　久保田 稔

NTT 光ネットワークシステム研究所

1　はじめに

我々は、将来の通信システムを支える抜本的なネットワークアーキテクチャに適用することを狙いとして、分散処理プラットフォーム DONA DPE[1] を検討している。これは、他のネットワーク事業者やサービス事業者との接続を考慮し、システムに高い相互接続性を持たせることを目的の1つとする。この相互接続性を達成する1つの方法として、分散処理環境の標準である OMG の CORBA[2] に基づいた ORB (Object Request Broker) (以下、DONA ORB) を実装している [3]。本稿では、通信サービスに要求されるリアルタイム性能の達成を容易にするため、通信方法を最適化したアプリケーションの記述を支援する方式を示す。

2　分散処理とリアルタイム性

今後、分散システムを構築する上では、標準化された ORB の採用が有力である。その理由は、(1) 分散透過性・相互接続性を利用して、柔軟なアプリケーションの構成が実現できることと、(2) 通信に関する機構をアプリケーションに対して隠蔽することにより、高いアプリケーションの記述性を達成できること、である。つまり、ORB を採用することにより、実行環境である ORB とアプリケーションの開発の分離が容易になる。

本稿で取り扱うリアルタイム性の保証とは、end-to-end の実行時間の保証とする。リアルタイム性を実現するためには、システム全体が連携して動作する必要がある。一方通信サービスは本質的に分散システム上に実現されるものであり、しかもリアルタイム性を保証することが求められる。通信サービスで要求されるリアルタイム性の特徴は、個々の時間制約は比較的緩いかわりに、同時に多数のサービスを時間制約を保証して実行する必要があることである。既存の通信網上のサービスではこのようなリアルタイム性を、下層の OS から上層のアプリケーションまで一括で開発することで、保証してきた。

A Facility to Assist Application Development on CORBA-compliant ORB
Takayuki Nakamura, Michiharu Takemoto, Satoshi Tanaka and Minoru Kubota
NTT Optical Network Systems Laboratories
3-9-11 Midori-cho, Musashino-city, Tokyo, 180-8585, Japan.
takayuki@ma.onlab.ntt.co.jp

将来の分散オブジェクト指向によるサービスでは、OS とアプリケーションの中間に位置する ORB がリアルタイム性のための機能を持つとともに、これらの間の橋渡しをすることが重要になる。

我々は ORB のリアルタイム化を実現する方法を、図1に示す3種類に分類する。本稿ではこれらの内、**(3) アプリケーションの通信最適化支援**を取り扱う。なお、**(2) ORB の処理軽量化**は DONA ORB をターゲットとして既に検討・実装した [3]。また **(1)RT-OS 機能**を用いた実行順序制御については、これを含む Realtime CORBA1.0 の仕様が OMG において制定中であり [4]、DONA ORB における検討結果が [5] で報告される。

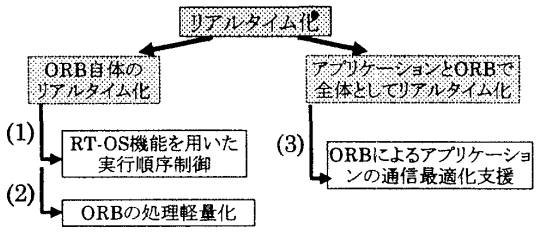

図 1: ORB リアルタイム化の3つの取り組み

3　アプリケーションの通信最適化支援

アプリケーションの通信最適化支援は、アプリケーションが通信方法・通信量を最適化して、より多数のクライアントがサービスを受けられるようにするための支援機能を、ORB が提供するものである。

前節で述べた通信サービスで要求されるリアルタイム性を向上させる方法として、キャッシュ・投機的実行を始めとする通信最適化アルゴリズムのアプリケーション自体への適用が挙げられる。これによりアプリケーションの通信量、処理量が削減されれば、一定の時間内に、より多くのクライアントに対してサービスを提供できるようになる。つまり、時間制約を保証しつつ多数のサービスを実行するリアルタイム性が向上する。

このような通信最適化アルゴリズムは、例えば現在のネットワーク上のサービスでは、Internet 上の WWW のキャッシュ技術 [6] などがある。これらの通信最適化アルゴリズムは、各種のアプリケーション内部に作り込まれる形で実装される。

同様の通信最適化アルゴリズムは将来の分散オブジェクト指向に基づく通信サービスにおいても、リアルタイム性の向上に有効である。しかし、アプリケーションの

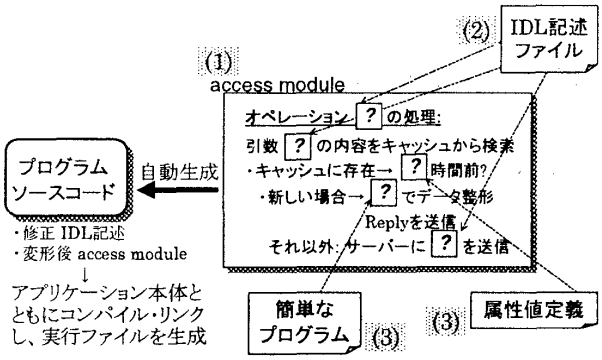

図 2: access module を用いるプログラムの生成手順

種類ごとに内部に通信最適化アルゴリズムを作り込むことは、分散オブジェクト指向の利点である記述性を損ね、将来の多様なアプリケーションの出現に支障をきたす。

本稿では、ORB 及び開発環境がアプリケーションの分離記述を支援し、通信方法を最適化したアプリケーションの記述を容易にする、以下の方式を提案する。

まず、最適化前のアプリケーションを作成する。これは、アプリケーションの処理の logic のみを記述した部分である。それとは独立に、キャッシュアルゴリズムなど通信方法を最適化するための部分である **access module** を作成する。両者を組み合わせることで、最適化アプリケーションが生成される。

Access module のライブラリ化による再利用が可能となり、アプリケーション作成者は本質的な logic 部分の記述に集中できる。Access module のみを変更することによって、データのプリフェッチ等、様々な通信最適化アルゴリズムを導入することも容易である。

4 access module の変形適合

個々の通信最適化アルゴリズムは、多くのアプリケーションに応用可能である。それらのアルゴリズムは、access module の形でライブラリ化して、複数のアプリケーションに適用できるのが望ましい。しかし、アプリケーションごとのインターフェース (syntax) 的、処理内容 (semantics) 的な差異が問題となり、そのままの形での適用は難しい。

そこで本稿では、access module に対してアプリケーションの syntax, semantics 情報を与え、access module を個々のアプリケーションに適合するよう変形する方式を提案する (図 **2**)。前処理系が (1)access module、(2)syntax 情報として IDL 記述ファイル、及び (3)semantics 情報として access module が参照する属性値及び短いコード断片を処理し、ソースコードを自動生成する。これらを logic 部分のソースコードと共にコンパイルすることで、目的とする最適化アプリケーションのプログラムを得る。

例として、情報表示サービスのクライアントにキャッシュ機能を追加する場合の記述を示す。図 **3**はデータのキャッシュ処理を実現する access module ライブラリ

```
#include "simple-cache.h"

#rewrite_method<POSTYPE, POS, DATTYPE, DAT, TIMESPAN, MODIFYFUNC>
CLASSNAME::METHODNAME(POSTYPE POS, DATTYPE##_out DAT), targetObj
{
  CacheEntry<DATTYPE> *ce;
  ce = theCache<DATTYPE>->lookup(POS);
  if (ce) {  // data exist.
    if (ce->updatedTime > curtime - TIMESPAN) {
      // return the cached data with a little modification.
      MODIFYFUNC(ce->content, curtime, DAT);
      return;
    }
  }
  targetObj->orig_##METHODNAME(POS, DAT);
  theCache<DATTYPE>->regcache(DAT, curtime);  // update the cache.
  return;
}
```

図 3: access module の記述例

属性値定義情報:

```
method WeatherForecast::ForecastTomorrow
  <Area_ptr, place, Image_out, result, 2*3600, retouchDetail>
```

コード断片:

```
void retouchDetail(Image_ptr img, time_t tim, Image_out &resultimg)
{
  ...  //天気情報の画像の細部(時刻表示)等を微調整する。
}
```

図 4: access module に与えるアプリケーション情報例

の記述例である。図 **4**はアプリケーションプログラマが記述する部分であり、これを前処理系に与えると図 3 中のキーワード部分がアプリケーションの指定する情報で埋められ、ソースコードが自動生成される。

この例において、既存のクライアント自体の記述を変更する必要はなく、追加記述分もわずかで良い。保持時間の指定や画像の微調整等、クライアントの内容依存の処理を記述でき、効果的なキャッシュ処理が実現できる。

5 まとめ

提案方式により、種々のアプリケーションに対して同一の access module を適用し、logic 部分の変更なしに通信方法を最適化できる。今後は access module の変形を行う簡単な検証システムを作成し、提案方式の有効性や課題を検証する。

参考文献

[1] Suzuki, S., Yamada, S., Kubota, M., Kogiku, I. and Matsuo, M.: DONA: Distributed Object-Oriented Network Architecture for Revolutionary Network Reconstruction, *XVI World Telecom Congress Proc. (ISS'97)*, Vol. II, pp. 459–465 (1997).

[2] OMG: The Common Object Request Broker: Architecture and Specification. Revision 2.2.

[3] 中村隆幸, 武本充治, 田中聡, 久保田稔: 分散オブジェクト実行環境の軽量化, 第 57 回情報処理学会全国大会論文集, 3F-03 (1998).

[4] OMG: Realtime CORBA 1.0 Request For Proposal. `ftp://ftp.omg.org/pub/docs/orbos/97-09-31.pdf`.

[5] 武本充治, 中村隆幸, 田中聡, 久保田稔: CORBA 準拠 ORB のリアルタイム化における緊急要求処理への適用可能性の評価, 第 58 回情報処理学会全国大会論文集, 2N-07 (1999).

[6] 藤浦豊徳, 内藤昭三: 投機的キャッシュ法における複数サーバ連携方式の検討, 第 57 回情報処理学会全国大会論文集, 5F-06 (1998).