CORBA による XML データの転送

5D-07

楯 武士　　山田 洋一
NTT コミュニケーションウェア(株)

1 はじめに

近年、利用が浸透しつつある CORBA(Common Object Request Broker Architecture)は、異機種分散環境上のオブジェクト技術として注目されている。また、XML(Extensible Markup Language)は、Web アプリケーション・EAI(Enterprise Application Integration)やドキュメント管理などの分野での利用が注目されている。

本報告は、CORBA システムでのデータ交換に XML を利用する手法を提案し、その評価について述べる。

2 背　景

CORBA でのデータの交換は、IDL で CORBA オブジェクトのインターフェースを記述し、クライアント及びサーバが CORBA オブジェクトをアクセスすることで行われる。しかし、サービスの追加が多いシステムでの CORBA の利用には以下の問題点が指摘されている。

1. **データの変更に伴う修正が大**:データ項目変更があった時、IDL ファイルを修正し、IDL ファイル・クライアント/サーバ AP をリコンパイルする必要があり、ファイル更新も同時に行う必要がある。
2. **データのポータビリティが低い**:他のシステムとのデータの交換を考えた場合、対象システムに合わせたデータのポーティングが必要となる。
3. **IDL 管理が煩雑**:複数のサーバからなるシステムの場合、それぞれのシステムごとに IDL を用意する必要があり、管理が煩雑になる。

これらの問題点を解決するために、本報告では CORBA システム上で XML を転送する手法を提案する。

3 XML を用いたデータ交換方式

CORBA のセキュリティやトランザクション管理などの利点を生かしつつ、前章の問題を解決する方法として、本報告では XML データを CORBA オブジェクトの変数として格納し、サーバに転送する方式を提案する(以下、この手法を「XML 転送方式」と呼ぶ)。以下に XML 転送方式の手順について示す(図 3-1)。

1. クライアントで入力データを XML データとして CORBA オブジェクト内に保存する。
2. サーバで XML のデータをパースし、DOM(Docuemnt Object Model)を生成する。
3. サーバは、DOM にアクセスすることで XML データを取得し、オペレーションを実行する。

また、XML データを転送する別の手法としては、クライアントで DOM オブジェクト生成した後、それを CORBA オブジェクトとして転送する方法も考えられる。しかし、DOM オブジェクトは一般にメモリ消費量が膨大になり、転送効率が悪化することが予想されたので、本報告では上記の方式を採用した。

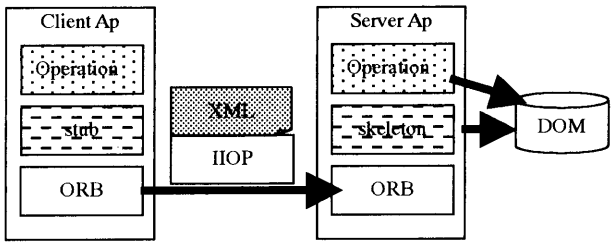


図 3-1 システム概略図

XML 転送方式によって、前章の問題点を次のように解決することができる。

1. **データの変更に伴う修正が不要** : データ項目に追加が生じても、それに伴う IDL ファイルの修正は生じない。これは従来 CORBA オブジェクトの各変数に格納していたデータを、XML に包含し転送することでデータの数・型を意識する必要がなくなったためである。
2. **データのポータビリティが向上** : XML は文字データから構成されているので他の独立したシステムとのデータのやり取りを行う場合に、データのポーティングを行うことなく、XML のデータを送信するだけで可能となる。
3. **IDL ファイルの共通化が可能** : すべてのデータを XML に包含することで、IDL ファイルに各データを定義する必要がなく単純な構造となる。その結果 IDL ファイルの汎用性が向上し、複数のサーバで共有することができる。

また、従来の方式に比べて、XML 転送方式には以下の問題点がある。

1. XML はデータにタグをつけることで、データの識別を行う。そのタグの分のデータ量が増大し、メモリ使用量、応答時間の悪化が懸念される。
2. CORBA は IDL でデータの型を規定することで型を厳格にチェックできるが、XML 転送方式は文字列データのみとなるので、CORBA の型のチェック機構を利用することができない。

4 評　価

前章のデータ転送量の増大がシステムに与える影響を調べるために、従来の方式と XML 転送方式をメモリ

XML Data Transformation on CORBA
Takeshi Tate, Youichi Yamada
NTT Communicationware Corporation

の使用量、1メッセージ当りの転送時間の点から比較評価を行う。今回の実験は、厳格な処理時間を要求されず、サーバのメモリ容量に比較的余裕があるシステムとしてWebアプリケーションを想定して評価を行った。

4.1 評価内容

今回の実験で用いたIDLを図4-1に、転送されるXMLデータを図4-2に示す。なお、従来方式で転送されるデータはこの中のタグで囲まれた部分(図4-2中で"123","John Doe"など)のみとなる。

```
// XML転送方式のIDL
interface Func1{
    attribute string xmlData;        XMLデータ
}
//従来方式のIDL
interface Func1 {
   attribute int id;
   attribute string name;
   attribute string email;           データ
}
```

図 4-1 本実験で用いたIDL

```
<?xml version="1.0"?>
<!DOCTYPEdepartment SYSTEM "department.dtd">
  <department>
     <employee>
       <id>123</id>                          データ部分
       <name>John Doe</name>
       <email>John.Doe@foo.com</email>
     </employee>
</department>
```

図 4-2 本実験で用いたXML

なお、今回の実験は下記の環境で実験を行った。

言語	Java(JDK1.2.1)(SunMicrosystems社)
ORB	Java IDL (SunMicrosystems社)
XMLパーサ	XML4J 2.0.15 (IBM社)

4.2 評価結果

図4-3に各方式のクライアント、サーバにおけるメモリの消費量を示し、図4-4に各方式の1メッセージ当たりの応答時間を示す(図4-3、4-4で、方式1はXML転送方式、方式2は従来方式を表す)。また、図4-3におけるメモリの使用量は、全メッセージが同時に転送されたときのメモリ消費量を示す。

4.3 評価結果の分析

- **メモリ消費量** : 図4-3からXML転送方式は、クライアントのメモリ消費量が従来方式と同程度であるのに対して、サーバのメモリ消費量は大幅に大きい。これはDOMによるメモリ消費が多量であるためである。

Java,JDKはSun Microsystems社の商標です

- **応答時間** : 図4-4からデータ転送量が6倍(従来方式の30Byte対して、XML転送方式は180Byte(図4-2))であるにも関らず、応答時間の差は3倍以内である。これはパーサによるXMLデータの解析やデータの転送よりも、CORBAオブジェクトへのアクセスにより時間が費やされるためである。

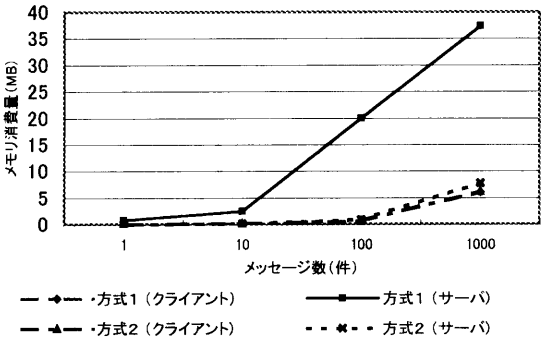
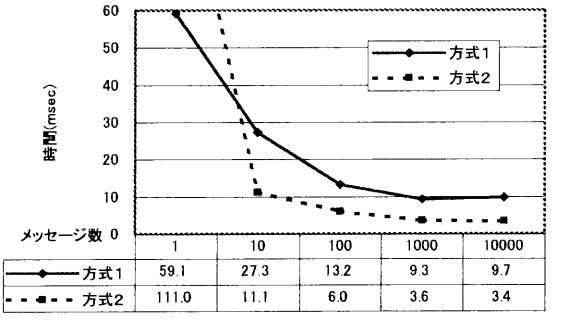


図 4-3 メモリ消費量



メッセージ数	1	10	100	1000	10000
方式1	59.1	27.3	13.2	9.3	9.7
方式2	111.0	11.1	6.0	3.6	3.4

図 4-4 1メッセージ当りの応答時間

5 考 察

3節で提起した問題点について考察する。

- データ転送量の増加によるパフォーマンス劣化は、4.3節から厳格な処理時間を求めずに、同時に発生するトランザクション数が限られたシステムにおいては影響が少ないと言える。
- CORBAの型チェック機構を利用することができないという問題については、現在W3Cで仕様化が進んでいるXML Schema(XMLのデータの型やデータ長を定義するための仕様)を利用し、XMLデータの型のチェックを行うことで解決が可能であると考える。

以上より、XML転送方式はWebアプリケーションなどへの適用が可能であると思われる。

6 まとめ

本報告は従来のCORBAによるC/Sシステムの問題を指摘し、それに対する解決方法としてXMLによるデータ交換であるXML転送方式を提案した。

また、XML転送方式と従来の転送方式の評価を行い、その適用範囲について言及した。