2J-04　　XMLを用いた分散オブジェクトのアクセス方式の実装と評価

堀内　浩規　　茂木　信二　　小田　稔周
（株）KDD研究所

1. はじめに

近年、分散しているアプリケーションやデータを統合化するプラットフォームとしてCORBA (Common Object Request Broker Architecture)[1]が普及しつつある。一方、インターネット上での電子商取引等でのデータ交換フォーマットとして、文書記述言語であるXML(Extensible Markup Language)[2]が注目されている。今後は、CORBAとXMLとの相互接続が重要になると考えられ、筆者等はCORBAサーバへのXMLによるアクセス方式を提案している[3]。本稿では、その方式に基いたゲートウェイを実装したので、その結果を述べる。

2. XMLによる分散オブジェクトアクセス方式[3]の概要

2.1 アクセス形態

本方式では、XML環境とCORBA環境との相互接続を実現するため、両者の手順を変換するゲートウェイを設ける(図1)。具体的には、CORBAクライアントがXMLのサーバを、XMLのクライアントがCORBAサーバをアクセス可能とするため、ゲートウェイはHTTP(Hypertext Transfer Protocol)を用いて転送されるXMLインスタンスとCORBAのIIOP(Internet Inter-ORB Protocol)の手順を変換する。

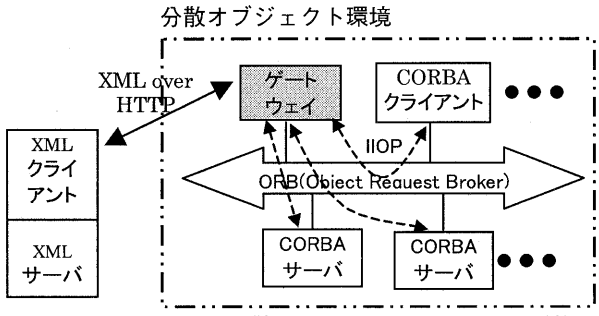


図1　XMLによる分散オブジェクトアクセスの形態

2.2 IDLからDTDへの対応付け

CORBAオブジェクトの属性取得･設定や操作と､XMLインスタンスとを対応付けるため、本方式では、オブジェクトのインタフェースを規定するIDL(インタフェース定義言語)とDTD(文書型定義)との対応付規則を規定する。図2にIDL定義とDTDとの対応付例を示す。ここでは、モジュール("Example")を文書に、その要素にインタフェース("Account")等)を定義する。インタフェースの要素は、オブジェクトの名前("cos_naming")、操作要求･応答("operation_ind")等)、属性取得・設定の要求･応答("attriute_get_ind")等)から構成する。さらに、操作の要素は、操作名、引数名、データ型名･値等の要素から構成する。CORBAのデータ型は、型を示すタグと値を示すPCDATAによるテキストとして表現する。

2.3 ゲートウェイの汎用性

ゲートウェイは、収容するCORBAサーバやXMLサーバの新規追加やIDLの変更に対しても、プログラムの変更を行う必要が無いように、CORBAの動的インタフェースを活用する。また、クライアント、サーバならびにゲートウェイにおいて、XMLインスタンスの作成・検証を容易とするため、IDLからDTDに自動変換するトランスレータを用意し、ゲートウェイは変換を行ったDTDを公開する。

Implementation and Evaluation of Access Method for Distributed Objects using XML
Hiroki HORIUCHI, Shinji MOTEGI and Toshikane ODA, KDD R&D Laboratories Inc.

```
module Example {
   interface Account {   //インタフェース
      attribute long account_num;   // 属性
      long deposit (in long amount);   // 操作
      long withdraw (in long amount);   // 操作 };
   interface Manager {   // インタフェース 省略 };
   // 他のインタフェース省略 };
```

(a) IDL定義例

```
<!ELEMENT Example ((Account | Manager | … )*>
<!ELEMENT Account (cos_naming ( operation_ind |
   attribute_get_ind | attribute_set_ind | operation_cnf |
   attribute_get_conf | attribute_set_conf )* )>
<!ELEMENT cos_naming ( reference | (naming_id,naming_kind?)+)>
<!ELEMENT operation_ind (operation_name, (in_param?, out_param?,
   in_out_param?, return_value? )*) >
<!ELEMENT attribute_get_ind (attribute _name, (long | … ))>
      中略
<!ELEMENT naming_id (#PCDATA)>
<!ELEMENT operation_name (#PCDATA)>
<!ELEMENT attribute_name (#PCDATA)>
<!ELEMENT in_param (type_name, ( long | … )>
      中略
<!ELEMENT type_name (#PCDATA)>
<!ELEMENT long (#PCDATA)>
```

(b) DTD対応付け例

図2　IDL定義とDTDとの対応付け

3. 実装と評価

3.1 システム構成

ゲートウェイのシステム構成を図3に示す。ゲートウェイの実装では、OSはWindows NT 4.0、XMLパーサはXML Parser for Java[4]、CORBAはOrbixWeb 3.2を使用し、Java Servletを用いたWebアプリケーションとして実装した。

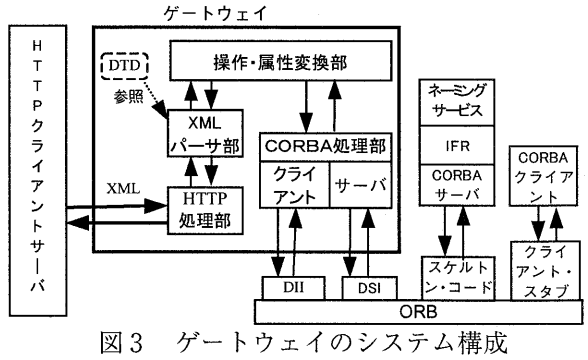


図3　ゲートウェイのシステム構成

HTTPクライアントからCORBAサーバにアクセスする際のゲートウェイの処理を以下に示す。

(1)HTTP処理部を経由してクライアントから受信したXMLインスタンスに対し、XMLパーサ部において、DTDを用いた検証とパースを行う。

(2)操作・属性変換部は、XMLパーサ部でエレメントハンドラ[4]が起動される毎に呼び出され、XMLインスタンス中に定義された操作やパラメータ等をCORBAサーバを呼び出す形式に変換する。

(3)CORBA処理部は、ネーミングサービスにより処理対象オブジェクトのトリファレンスを取得する。

(4)さらに、IFR(インタフェース・リポジトリ)を参照して、パラメータ設定等に必要なデータを補った後、DII(動的起動インタフェース)を呼び出して、CORBAサーバに操作要求を送信する。

(5)CORBA処理部によりCORBAサーバから操作応答を受信し、操作・属性変換部により変換とDOM(Documet Object Model)の値設定を行った後、XMLパーサ部でDOMをXMLインスタンスに変換する。

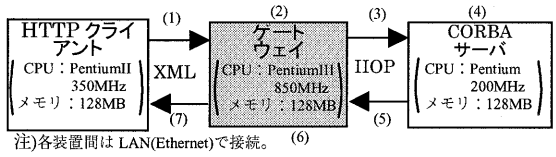
(6)HTTP処理部は上記(5)で作成したXMLインスタンス

を HTTP サーバに送信する。

CORBA クライアントから XML サーバにアクセスする際のゲートウェイの処理は、基本的に上記の逆の処理となる。即ち、DSI(動的スケルトンインタフェース)を経由して、CORBA 処理部が CORBA の操作要求を受信し、操作・属性変換部で DOM に値を設定した後、XML パース部で XML インスタンスを作成し、HTTP 処理部によりそれを HTTP クライアントに送信する処理等を行う。

3. 2 性能評価

実装したゲートウェイの性能を評価するため、応答時間とゲートウェイの処理時間を測定した。図 4 に試験構成を、表 1 に測定結果を示す。CORBA サーバへの操作発行では、単一操作発行の場合と、XML インスタンスに複数操作を定義し、操作を逐次発行する場合とを計測した。なお、CORBA サーバでは、単純に値を返送する処理のみを実現している。



注)各装置間は LAN(Ethernet)で接続。

図 4 試験構成と観測点

表1 応答時間と処理時間の測定結果

測定項目[注1]		応答	ゲートウェイ処理時間[注3]		
操作数	引数型	時間[注2]	全体	パース	変換
1	long	170.2ms	78.6ms	13.7ms	33.1ms
1	String	180.9ms	94.2ms	25.3ms	23.5ms
1	Struct	187.5ms	98.5ms	32.9ms	46.7ms
2	long	334.0ms	122.8ms	22.7ms	46.5ms
3	long	464.1ms	145.1ms	41.3ms	58.1ms
4	long	591.3ms	201.2ms	46.6ms	71.3ms

注1) 全項目で引数の個数は 2 個。String 型の文字列の長さは 8 バイト
Struct 型のメンバは、long 型と String 型の 2 つのメンバからなる。
注 2) 図 3 (1)~(7)の合計。注 3) 図 3 (2)と(6)の合計。

表 1 より、一操作あたりのゲートウェイの処理時間は 78.6～98.5ms 程度で、また、XML インスタンスや CORBA の送信時間等を含めた応答時間でも 170.2～187.5ms 程度のオーバーヘッドであり実用的な性能を達成している。また、複数操作の逐次処理では、一操作増える毎に約 130ms ほど増加する。

4. ゲートウェイの適用形態

実装したゲートウェイの適用領域を明確化するため、3 種類の適用形態を以下に示す。

4. 1 XML と CORBA システムとの相互接続(図 5(a))

本形態は、CORBA を用いてデータ交換を行う既存システム(A 社と B 社)と XML によりデータ交換を行う既存システム(C 社と D 社)において、ゲートウェイを介した相互接続を可能とする形態である。XML クライアントは、CORBA サーバの IDL に対応した XML インスタンスを送受信することにより、CORBA サーバにアクセスすることが可能となる。

4. 2 通信サービスのアクセスへの適用(図 5(b))

WWWを用いたインターネット経由の通信サービスの実現方法として、WWWサーバのバックエンドにCORBAを適用する形態がある[5]。ここでは、端末PC側(図5(b)の左下の端末)でアプレットをダウンロードし、ブラウザ上でCORBAクライアント・サーバを動作させ、バックエンドのアプリケーションへの直接アクセスや、バックエンド側からのプログラムの起動等により、高度で効率的なサービスの提供が期待できる。しかしながら、この形態では、以下のような問題点がある。

[問題1] ファイアウォールでは、HTTPのポートを開放している場合が一般的であるが、IIOPのポートを開放していない場合が多く、通過が困難な場合がある。

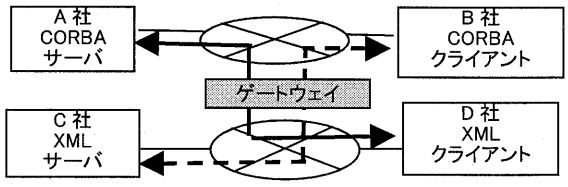
[問題 2] 実行形式のプログラムに ORB や AWT(Abstract Window Toolkit)等が含まれ、必要なメモリ量が増大する。このため、携帯端末や PDA 等の処理能力の低い端末では実行が困難な場合がある。

[問題3] 上記[問題 2]と関連して、ダウンロードするアプレットのプログラムが比較的大きく、アクセス回線が細い場合等にはダウンロード時間が増大する。

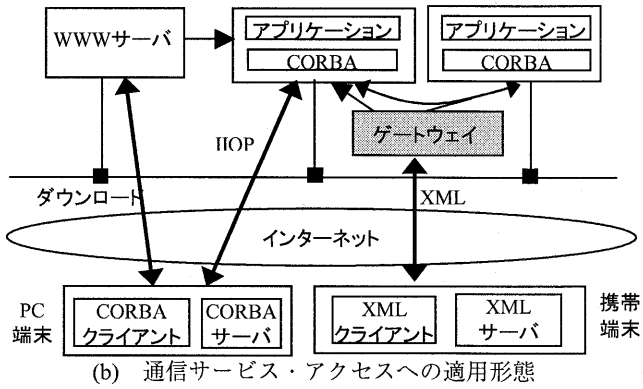
WWW と CORBA 連携の利点を生かしつつ、上記の問題点を解決する方法として、ゲートウェイの適用が考えられる(図 5(b)の右下の端末)。端末側では、ダウンロードが不要のみならず、XML に基づくテキスト処理により、通信サービスにアクセス可能となるため、処理負荷が軽減される。さらに、HTTP の利用により、ファイアーウォールの通過も容易になる。

4. 3 CORBA システム間の相互接続への適用(図 5(c))

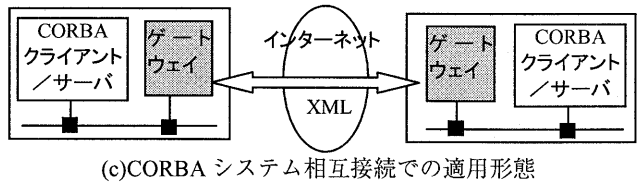
インターネットを介した CORBA システム間の接続では、上記 4.2 節の[問題 1]で述べたファイアーウォールの問題により、相互接続が困難な場合がある。そこで、本ゲートウェイを両端のシステムに適用して、インターネット上での相互接続手順として XML over HTTP を利用することにより、相互接続が容易になる。



(a) XML と CORBA システム相互接続の適用形態



(b) 通信サービス・アクセスへの適用形態



(c)CORBA システム相互接続での適用形態

図 5 ゲートウェイの適用形態

5. おわりに

本稿では、XML を用いた分散オブジェクトの アクセス方式に基づくゲートウェイの実装と評価結果を報告した。性能評価を通してゲートウェイの実用性を示すとともに、3 種類の適用形態を示して有効性を示した。最後に、日頃御指導頂く(株)KDD 研究所 秋葉所長、浅見副所長、松島副所長ならびに小花取締役に感謝します。

参考文献：

[1]:Object Management Group, "The Common Object Request Broker: Architecture and Specification Rev. 2.2", 1998. http://www.omg.org/.
[2]:W3C,"Extensible Markup Language (XML) 1.0", 1998.
[3]:堀内, 茂木, 小田, "XML を用いた分散オブジェクトのアクセス方式", 6S-05, 第 60 回情処全大, Mar. 2000.
[4]:IBMCorporation, "XML Parser for Java", 1998
[5]:S. Motegi, et al., "Design and Evaluation of Computer Telephony Service in a Distributed Processing Environment", IEICE Trans. on Commun. Vol. E83-B, No.5, pp.1075-1084,2000.