システムソフトウェアと
オペレーティング・システム 76－12
（１９９７．８．２１）

分散 Java 実行系 HORB の基本性能の評価

平野 聡
hirano@etl.go.jp

HORB Open Consortium　電子技術総合研究所
http://ring.etl.go.jp/openlab/horb/

HORB は Java を分散オブジェクト指向言語に拡張する Java 用の ORB(オブジェクト間通信機構)である。本論文では HORB と他の分散オブジェクト技術(RMI, CORBA IIOP)の基本性能を比較、評価した。評価の結果、HORB はリモートオブジェクトの生成やリモートメソッドの呼び出しは大変高速であるにもかかわらず、オブジェクト配列の転送の性能が低いことが明らかになった。そこでシリアライザをキャッシュする等の改良を施したところ、オブジェクト配列の転送の性能は大幅に改善され、他の分散オブジェクト技術より高い性能を示すようになった。

Preliminary Performance Evaluation of a Distributed Java: HORB

HIRANO Satoshi
hirano@etl.go.jp

HORB Open Consortium　Electrotechnical Laboratory
http://ring.etl.go.jp/openlab/horb/

HORB is an ORB (Object Request Broker) for Java that extends Java as a distributed object oriented language. In this paper, the performance of basic operations of HORB is compared and evaluated with the performance of other distributed object technologies including RMI and CORBA IIOP. While the results showed the performance of remote object creation and remote method call outperformed other technologies, the performance of transferring arrays of objects tended low. After improving the runtime by adopting a serializer cache, the performance was much improved.

1 はじめに

1995年にSun社によってネットワークポータブルなオブジェクト指向言語 Java[1]が発表されて以来、1995 年電総研による分散 Java 実行系 HORB[2,3,4] の公開を先駆けに、1996 年 CORBA2 のインターネット ORB 間共通プロトコル IIOP[6]の Java 上での実装、Microsoft 社による Windows での Distributed COM[7]の実装、Sun 社による Java 用オブジェクト間通信機構 JavaRMI の API[8]の公開と続き、Java を中心とする分散オブジェクト技術が相次いで製品化された。現在、これらの分散オブジェクト技術を用いたシステム開発が一般的になりつつあり[9,10]、分散オブジェクト技術によるネットワークコンピューティングは実用期に入ったと言える。

本論文は HORB の基本性能を測定し、他の Java 用分散オブジェクト技術と比較することにより HORB の性能の特性を調査し、必要に応じて HORB の処理系を改良して性能向上を図る事を目的とする。
以下、2 章で代表的な Java 用分散オブジェクト技術の概要について述べたのち、3 章で HORB, RMI, CORBA IIOP についてそれぞれリモートオブジェクトの生成、リモートメソッドの呼び出し、リモートオブジェクト呼び出しにおけるクラス継承、オブジェクト転送等の分散オブジェクト技術における基本的な性能を評価する。性能評価の結果、HORB はリモートオブジェクトの生成やリモートメソッドの呼び出しは大変高速であるにもかかわらず、オブジェクト配列の転送の性能が低いことが明らかになった。3 章の後半では HORB のオブジェクト配列の転送の性能を向上させるための改良について述べ、改良後の性能を評価する。改良によってオブジェクト配列の転送は最大 17 倍高速化された。

2 分散オブジェクト技術の概要

本章では本論文で評価を行う代表的な分散オブジェクト技術の概要について述べる。分散オブジェクトを言語のレベルで実現する方法を大きく分けると、

1. 既存のプログラミング言語を拡張したり、新たな分散オブジェクト指向言語を定義する言語拡張の方法(例：C++を拡張して分散 XX 言語を実現する)
2. 既存のプログラミング言語とオブジェクト間通信機構(ORB, Object Request Broker)を組み合わせて分散オブジェクトを実現する方法(例：Java と CORBA を使用する)

の 2 種類のアプローチがある。言語拡張の方法は分散オブジェクトを利用するための言語機構が言語にシームレスに組み込まれているためプログラマの負担は少ない。しかし、新たな言語、あるいは言語の方現を広く普及させることは難しいため、多くの言語拡張が開発されてきたにもかかわらず、分散言語が一般的となるには至っていない。
既存の言語に ORB を組み合わせて利用する方法は、既に普及している言語処理系や開発環境を利用できる利点はあるものの、言語のオブジェクトのセマンティクスと ORB の提供するオブジェクトのセマンティクスが異なっているとプログラマに負担を強いたり、ORB を利用するためにプログラム上の制限が存在する等の欠点がある。ネットワークコンピューティングでよく用いられる Java の場合、ORB には

- 言語に依存しない汎用 ORB の Java バインディングを用いる
- Java 専用の ORB

の 2 種類がある。本論文では Java と Java 専用 ORB 及び汎用 ORB を用いる分散オブジェクト技術について評価・比較する。

2.1 HORB

HORB は上記 1, 2 の利点を同時に満たすため、Java を分散オブジェクト指向言語としてシームレスに拡張することを目標とする Java 専用の ORB である[2,3,4]。

```
class Server {
  Video play(String title) {...}
}
```

リスト 1 HORB のリモートオブジェクト

```
Server_Proxy s = new Server_Proxy
                 ("horb://www.etl.go.jp/");
video = s.play("Apollo 13");
```

リスト 2 リモートオブジェクトの生成と呼び出し

リスト 1 の Server クラスを www.etl.go.jp というマシン上にリモートオブジェクトとして生成し play()メソッドを呼び出す処理はリスト 2 のごとく簡単な記述となる。リスト 2 において、Server クラスの代わりに、リモートオブジェクトを生成するホスト名を URL で指定して Server_Proxy クラスを new する点が非分散プロ

グラミングとの僅かな差(おまじない)である。

HORB は以下の特徴を有する。

- プログラミングが大変容易である。Java の言語仕様を変更していない。Java のプログラムに僅かの「おまじない」を加えるだけで分散実行が可能となる。
- プログラミング上の制約が少ない。たとえばリモートオブジェクトは特定のクラスを継承する必要がない。デザインパターン内の任意の位置にリモートオブジェクトを配置可能である。
- 処理系と実行系が JDK1.0 レベルの Java で記述されているため、どんな Java 実行系上でも動作する。

以下に主な分散オブジェクト機能について述べる。

リモートオブジェクトの動的生成: クライアントマシン上のオブジェクト(アプレットを含む。以下、単にクライアントと記す)からサーバマシン上にリモートオブジェクトを新しく生成する。

リモートオブジェクトへの接続: クライアントからサーバ上に既に存在するリモートオブジェクトに接続する。オブジェクトにはクライアントに対応するスレッドが生成され並行実行するため、サーバは複数のクライアントに同時にサービスを行うことが可能である。

リモートメソッド呼び出し: クライアントからリモートオブジェクトのメソッドを呼び出す(同期または非同期)。

オブジェクト転送: メソッドの呼び出しの引数及び返値は、任意の基本型、オブジェクト、配列である。オブジェクト転送では、リモートオブジェクトへの参照は参照渡し、それ以外は値渡しとなる。構造を有するオブジェクトはオブジェクトの参照関係が到達する範囲で、参照関係の構造を保ったまま転送される。現在のバージョンでは private 変数の転送はできない。また、転送されるクラスはオブジェクトのシリアライザを含む Proxy クラスが必要である。

他に、分散ガベージコレクションや、クラス中の特定のメソッドにアクセス制限を加えたり、オブジェクトが転送される際に特定のメソッドを自動的に呼び出す(hooks)など、多様なプログラミング機能を有する。また、分散システム構築フレームワークとして分散アクセスコントロールリストによるセキュリティ、永続オブジェクト、分散オブジェクト管理、WWW との統合といった機能を有する。多くのベンダーが提供する Java の API やコンポーネントを併用することで、高度な分散システムを構築することが可能である。

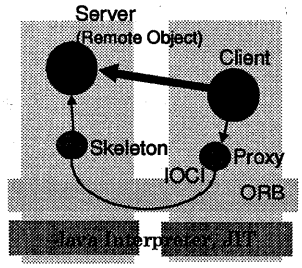

図 1 Proxy による分散オブジェクトの実現

HORB では分散オブジェクトの実現法として、Proxy オブジェクト[5]による方式をとる(図 1)。クライアントのオブジェクトは、リモートオブジェクトの代理としてローカルな Proxy オブジェクトをアクセスする。Proxy オブジェクトは通信処理を行う ORB を介してリモートオブジェクトを利用することにより、リモートオブジェクトと同等のメソッドをクライアントに提供する。Proxy と Skeleton は HORBC コンパイラによって生成される。

2.2 JavaRMI

RMI(Remote Method Invocation)は Sun 社が定めた Java 用 ORB の API[8]である。JDK1.1 には RMI の実装が含まれている。RMI はリモートオブジェクトへの接続、リモートメソッドの呼び出し、オブジェクト転送といった基本的な分散オブジェクト機能を定義する。

RMI ではリモートオブジェクトの定義を interface として記述する。リモートオブジェクトは Remote インターフェース、UnicastRemote Object クラス、Serializeable インターフェース等を継承する必要があり、Java とのシームレスな一体化を妨げている。また、デザインパターン中でリモートオブジェクトを配置できる位置が限定される。

2.3 CORBA

CORBA[6]は OMG(Object Management Group)が定めた分散オブジェクトアーキテクチャの規約であり、以下の特徴がある。

- 言語や機種に依存しない。
- リモートオブジェクトの定義は OMG IDL 言語で記述する。
- リモートオブジェクトへの接続、リモートメ

ソッドの呼び出し、参照渡しによるオブジェクト転送といった基本機能と、CORBA Servicesによる拡張機能を提供する。
- IIOP(Internet Inter-ORB Protocol)プロトコル[6]を用いることにより、ベンダー間の相互運用性が実現される。

多くの CORBA の実装ではリモートオブジェクトを実装するクラスは Skeleton を継承する必要がある。RMI と同様にデザインパターン中でリモートオブジェクトを配置できる位置が限定される。

3 基本性能の評価と改良

本章では2章で述べた各分散オブジェクト技術の基本性能を評価する。性能評価は以下の点について行う。
- リモートオブジェクトの生成・接続
- リモートメソッド呼び出し
- オブジェクト転送

評価環境は以下の通りである。

- Pentium Pro 200MHz 64MB メモリの計算機2台
- OS は Windows NT 4.0 SP3
- ネットワークは 100Mbps と TCP/IP (100 Base-TX スイッチングハブ)
- HORB は 1.3.b1、RMI は JDK1.1.2、CORBA は IONA 社の OrbixWeb 2.0.1 (Java による IIOP の実装)を使用し、JDK1.1.2 の Java インタプリタで実行する

3.1 リモートオブジェクトへの接続とリモートオブジェクトの動的生成

最初は、サーバ上にリモートオブジェクトを登録しておき、クライアントからそのオブジェクトに接続する際の性能を評価する。図2に実行時間の測定結果を示す。HORB と RMI はほぼ同等の性能を有し、Orbix IIOP はその倍の実行時間を要した。
次に、クライアントからサーバ上にリモートオブジェクトを動的に生成する際の性能を評価する。HORB はリモートオブジェクトを動的に生成する機能を有しているが、RMI と CORBA の場合はそのような機能を有していないため、予め存在するリモートオブジェクトに接続し、そのオブジェクト内で新オブジェクトを生成して、リモートオブジェクトリファレンスとして戻してもらう2段階の操作が必要となる。図3に実行時間を示す。HORB のリモートオブジェクトの動的生成機能の効果は高く、RMI や Orbix IIOP の6、7倍の性能が得られた。

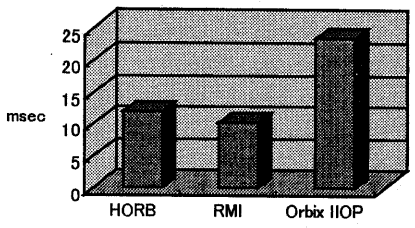

図2 リモートオブジェクトへの接続

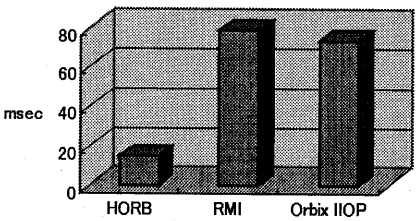

図3 リモートオブジェクトの生成

3.2 リモートメソッド呼び出し

リモートオブジェクトのメソッドを呼び出す評価には以下のようなメソッドを用いた。

```
Int methodA1(int a1);
Int methodA2(int a1, int a2);
....
Int methodA6(int a1, int a2, ..., int a6);
```

リスト 3 リモートメソッドの定義

＊リモートメソッドは引数としていくつかの整数を受け取り、返値として整数を返す。CORBA の場合は int 型がないため、long 型を用いた(Java の int 型にマップされる)。また、比較のため、同様の処理を行う Java のソケットによる実装も評

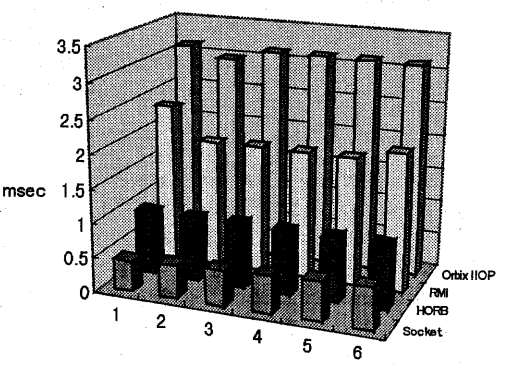

図4 リモートメソッド呼び出し

価する。
図4に結果を示す。横軸は渡す引数の数である。ソケットによる実装以外は引数の数の影響が現れてこないことがわかる。ソケットとHORBを比較するとHORBの方が約1.7倍の実行時間を要した。HORBとRMIを比較するとHORBは約2倍、HORBとOrbix IIOPを比較するとHORBは約3.3倍の性能を示すことが明らかになった。

3.2 継承をしているリモートオブジェクトへのメソッド呼び出し

リモートオブジェクトがクラス継承している場合のメソッド呼び出しの性能を評価するため、以下のようなクラスを用意した。クラスI5は深さ5レベルの継承をしており、クラスI4は深さ4レベルの継承をしている。クラスI0はRMI, CORBAの場合はSkeleton等を継承する。各クラスは各々10個のメソッドを含んでいる。

```
class I0 {...}
class I1 extends I0 {...}
class I2 extends I1 {...}
class I3 extends I2 {...}
class I4 extends I3 {...}
class I5 extends I4 {...}
```

リスト4　リモートオブジェクトの継承関係

図5に結果を示す。横軸はリモートオブジェクトの継承の深さである。例えば、5レベル深さの場合クラスI5のオブジェクトのクラスI0のメソッドを呼び出している。RMIとOrbix IIOPの場合、Proxy(Stub)クラスは上位クラスのメソッドすべてに対するスタブメソッドを含んでいる。メソッドの探索はProxyクラス内のサーチとなる。HORBの場合、Proxyクラスは対応するクラスのメソッドのスタブのみを含み、継承は上位のProxyクラスを継承することにより実現している。探索はクラスの探索とクラス内でのメソッドの探索の順となる[3]。ひとつのクラスが含むメソッドの数をS、継承の深さをI(リモートメソッドの総数はS*I)とすると、リモートメソッド探索のための比較の回数の期待値はRMIとOrbix IIOPの場合(S*I)/2であるのに対し、HORBの場合(I+S)/2となり、継承の高速化が図られている。図5でもOrbix IIOPが継承の深さが増えるに従って実行時間も増加しているのに対し、HORBでは影響が見られないことがわかる。

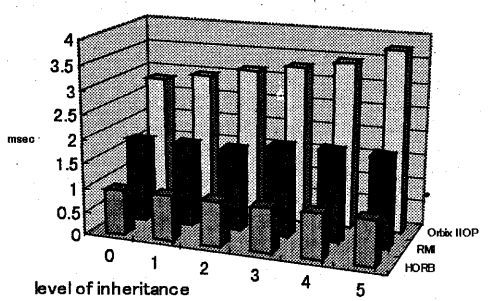

図5　継承しているオブジェクトの呼び出し

3.3 オブジェクト配列の転送

オブジェクト配列の転送は行列の演算などでよく用いられるであろう。そこで、オブジェクト転送の評価にはリスト5に示す整数をひとつ含むクラスと、そのクラスの不定長配列を引数として受け取るリモートメソッドを用いた。

```
class Data { int a; }
class Data2 extends Data {}
class Server {
    void method(Data[] array) {}
}
```

リスト5　オブジェクト転送のためのクラス

図6に結果を示す。横軸はmethod()に渡す配列内のオブジェクトの数である。現在のCORBAではオブジェクトを値渡しすることができないため、引数はstructのシーケンスとして渡した。従って、Orbix IIOPは実行時の型チェックが不要であるため、最も高い性能を示した。一方、HORBは最も低い性能を示した。HORBではリモートメソッドのシグニチャに記述されたクラスと実際に転送されるオブジェクトのクラスが同一である場合、クラス名を転送しない最適化を行っているが、まだオーバーヘッドが大きい。RMIはJava実行系に内蔵されたシリアライザを使っているため高速である。
図7にDataの配列の代わりにDataを継承するサブクラスData2の配列を引数として渡した場合の結果を示す。この場合、実行時に実際のクラス名も転送する必要がある。HORBの性能低下は著しい。

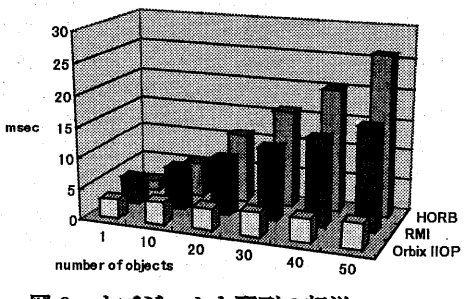

図6　オブジェクト配列の転送

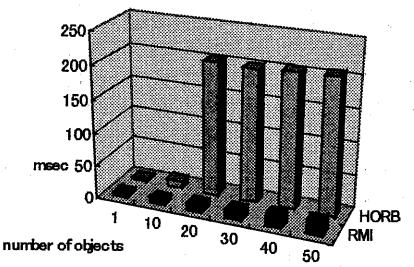

図7 サブクラスを転送した場合

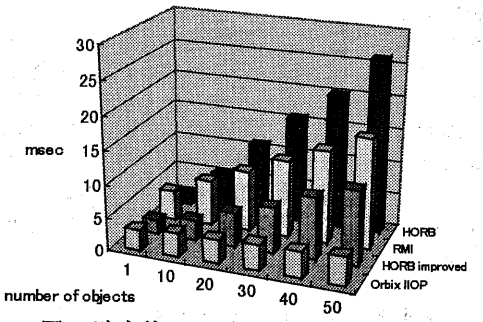

図8 改良後のオブジェクト配列の転送

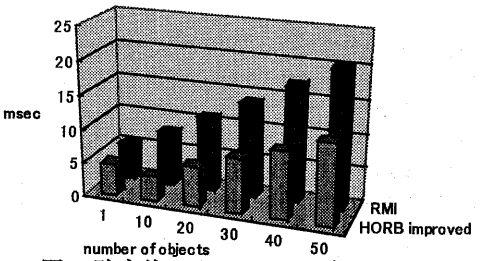

図9 改良後のサブクラスの転送

3.4 オブジェクト配列の転送の効率化

前節で見られた性能低下を改善するため、HORBの実行系に以下の改良を施した。

1. HORB ではオブジェクトを転送する際、そのオブジェクトのシリアライザを内蔵するProxy クラスをインスタンス化してシリアライザを呼び出している。Proxy のインスタンス化のコストを削減するため、Proxy のキャッシュを導入する。即ち、同じクラスのオブジェクトが連続して転送される場合には、毎回そのクラスの Proxy をインスタンス化せずに同じ Proxy を再利用する。
2. 転送されるオブジェクトのクラス名を送らなくてはならない場合でも、前回と同じクラス名であった場合は「前回送ったクラスと同じクラスのオブジェクト」であるというタグを送信することでクラス名の転送回数を削減する。

図8、図9に改良後の結果を示す。それぞれ図6、図7に対応しており、改良後の測定値は HORB improved として示している。改良により、いずれの場合も大幅な改善が見られた。

4 おわりに

本論文では分散Java実行系HORBと他の分散オブジェクト技術の基本性能を評価した。評価の結果、HORB はリモートオブジェクトの生成やリモートメソッドの呼び出しは大変高速であるにもかかわらず、オブジェクト配列の転送の性能が低いことが明らかになった。そこでシリアライザをキャッシュする等の改良を施したところ、オブジェクト配列の転送の性能は大幅に改善された。今後は更に高速化を図るとともに、Java の内臓シリアライザも利用可能にする予定である。HORBの開発はオープンである。研究、開発に利用して頂くことを希望している。

謝辞 議論、改良を行っている HORB Open Consortium、horb メーリングリストの参加者に感謝する。

参考文献

[1] Sun Microsystems, The Java language specification, 1996
[2] 平野, HORB: ワールドプログラミングのための分散並列オブジェクト指向言語, WOOC'96, 1996
[3] 平野, 分散Java実行のためのポータブルなORBの構成法, 情報処理研究報告96-OS-73, pp55-60,1996.
[4] HIRANO, S., HORB: Distributed Execution of Java Programs. World Wide Computing and Its Applications, 1997.
[5] Shapiro, M., Structure and Encapsulation in distributed Systems: The Proxy Principle, ICDCS, pp.198-205, 1986
[6] Object Management Group, CORBA2 Universal Networked Objects, 1995
[7] Brown, N., Kindel, C., Distributed Component Object Model Protocol -- DCOM/1.0, Internet Draft, 1996
[8] Sun Microsystems, Remote Method Invocation Specification, 1997
[9] Duan, N., Distributed Database Access in a Corporate Environment Using Java, 5th Int.WWW Conference, 1996
[10] Yamanaka, A., Nakajima, S., Tomono, M., Tonouchi, T., A HORB-Based Network Management System, ICODP'97, 1997
[11] Gamma, E., Helm, R., Johnson, R., Vlissides, J., Design Patterns, Addison-Wesley, 1995