遠隔メソッド起動におけるシリアライズの最適化の一手法

2N－4

日比野啓† 河野健二† 益田隆司†

†東京大学大学院 理学系研究科 情報科学専攻

1 はじめに

遠隔メソッド起動(RMI)とは，遠隔オブジェクトのメソッドであってもローカルなメソッド呼び出しと同じシンタックスで呼び出す技術であり，CORBA[3]やJava RMI[1]など多くの分散オブジェクト・システムの基盤技術となっている．RMIが分散システムの基盤技術として広く利用されている一因は，オブジェクトの相互運用性が保証しやすい点にある．オブジェクトの相互運用性とは，利用している言語処理系や計算機のアーキテクチャによらず，オブジェクトの受け渡しが可能であることをいう．

RMIでは，オブジェクトの相互運用性を保証するために，オブジェクトを受け渡す際にシリアライズと呼ばれる処理を行う．オブジェクトのシリアライズとは，リファレンスによる参照構造をもつオブジェクトの構造を整列化して転送し，受け手側で(受け手の)メモリ・レイアウトにあわせてオブジェクトの構造を再構成する処理をいう．オブジェクトのシリアライズは，オブジェクトの相互運用性を保証するための鍵となる技術であるが，オブジェクトの整列化／再構成のために頻繁にメモリ参照を行うため，RMIによる遅延時間のかなり大きな部分を占めることが知られている[2].

本稿では，RMIの最適化手法のひとつとして，オブジェクトのシリアライズを最適化する手法を提案する．提案方式の特徴は，通信先の計算機のアーキテクチャを知り得た場合に，そのアーキテクチャで直接利用可能なオブジェクトを転送している点にある．本稿ではこの最適化による実行時性能の向上について報告する．

2 相互運用性実現のためのシリアライズのオーバーヘッド

分散環境を構築する場合，既存の分散環境に新しいアーキテクチャの計算機を追加する形をとることが多い．そのため，オブジェクトの相互運用性を保証するには，メモリ・レイアウトの異なるアーキテクチャ間でオブジェクトの受け渡しが可能である必要がある．オブジェクトのシリアライズには，メモリ・レイアウトの変換を行うタイミングによって，(1)送り手と受け手の両方で変換，(2)受け手のみが変換，(3)送り手のみが変換，の三つの方針がある．

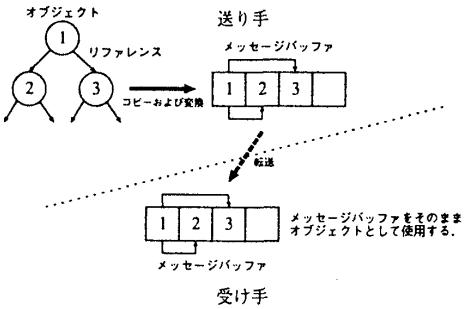

図1:「送り手のみが変換」の方式

「送り手と受け手の両方で変換」の方式では，送り手側の計算機で，XDRなどのアーキテクチャに依存しないデータの正規表現にオブジェクトを変換してから転送する．受け手の側では，データの正規表現からオブジェクトの構造を再構成する．この方式の利点は，新たなアーキテクチャを持つ計算機を追加した場合でも，そのアーキテクチャとデータの正規表現の間でのデータ変換を行うルーチンのみを提供すればよく，相互運用性を保証しやすい点にある．しかし，送り手側と受け手側の両者でオブジェクトのコピーおよびデータ変換を行う必要があり，そのオーバヘッドが大きい．

「受け手のみが変換」の方式では，送り手側はデータの内部表現は変換せずに，単にオブジェクトを通信用のバッファに複製して転送する．受け手側でオブジェクトの構造を再構成する際に，メモリ・レイアウトの変換を行う．この方式では送り手側でコピーが1回，受け手側でコピーとデータ変換が1回行われる．新たなアーキテクチャを持つ計算機を追加した場合には，その計算機と通信しうるすべての計算機と追加したアーキテクチャのメモリ・レイアウト間のデータ変換ルーチンを付加する必要がある．

「送り手のみが変換」の方式では，送り手側で受け手側の計算機のメモリ・レイアウトにオブジェクトを変換してから転送する．そのため，受け手側では転送されたデータをそのままオブジェクトとして利用できる．したがって，送り手側でコピーとデータ変換を1回行うだけでよい(図1)．しかし，「送り手側のみ変換」の方式と同様に，新たなアーキテクチャを持つ計算機を追加した場合には，すべての計算機にメモリ・レイアウトの変換ルーチンを付加する必要がある．これら三つの手法の得失を表1にまとめる．

Kei Hibino, Kenji Kono and Takashi Masuda.
Dept. of Information Science, University of Tokyo

表 1: シリアライズの方式の比較

	両方で変換	受け手のみ	送り手のみ
相互運用性	○	×	×
コピーの回数	2	2	1
データ変換の回数	2	1	1

3 Hybrid Serialization

本稿では，2節で述べた三つの方式のうち最も相互運用性を保証しやすい「送り手と受け手の両方で変換」と，最も実行時オーバーヘッドの小さい「送り手のみが変換」の方式を組み合わせて用いることを提案する．提案方式のねらいは，RMI の持つ相互運用性を損なうことなくシリアライズのオーバーヘッドを低減させることにある．

従来の RMI では遠隔オブジェクトのバインド時にはオブジェクトの位置情報だけが受け渡されるが，本稿で想定する RMI システムでは，位置情報だけでなく通信相手のアーキテクチャを表す識別子も受け渡されるものとする．このアーキテクチャの識別子から，そのアーキテクチャにあった変換ルーチンをオブジェクトの送り手側が持っているときには「送り手のみが変換」の方式を用い，そうでないときには，従来通り「送り手と受け手の両方で変換」の方式を用いる．

「送り手のみが変換」の方式では，送り手側で受け手側のメモリ・レイアウトに合わせてオブジェクトを変換する．このとき，オブジェクト間のリファレンスも受け手側のアドレス空間で有効な仮想アドレスに変換しておかねばならなず，受け手側の通信用バッファのアドレスが必要となってしまう．しかし，RMI のたびに受け手側からアドレスを通知してもらうのでは通信回数が増大し，非現実的である．この問題を回避するために，われわれの実装では，送り手側ではリファレンスを通信用バッファ内でのオフセットに変換しておき，受け手側で通信用バッファの先頭アドレスをオフセットに足しこんでリファレンスを補正する方式を採用した．

4 実験

「送り手のみが変換」の方式を用いることによって，従来の「送り手と受け手の両方で変換」の方式に比べ，シリアライズのコストをどの程度低減できるのか，実験を行なった．実験には各ノードが 2 つの整数フィールドをもつ二分木を用いた．ネットワークには 10Mbps の Ethernet を用いた．異機種間でのシリアライズのコストを計測するため，計算機には以下の 2 機種を用いた．

CPU	主記憶容量	OS
UltraSPARC 200MHz	160Mbyte	Solaris 2.5.1
Pentium II 400MHz	128Mbyte	Linux 2.0.35

異機種間でのシリアライズのコストと同機種間でのシリアライズのコストを測定した．その結果をそれぞれ図 2と図 3に示す．図中の XDR は，送り手側で XDR 表現にオブジェクトを変換し受け手でオブジェクトを再構成する方式であり，図中の「XDR 部分評価」は，Muller[2] らによって提案されている XDR ルーチンに部分評価を行なう方式であり，図中の sender-only が「送り手のみが変換」の方式である．これらのグラフから，「送り手のみが変換」の方式は，XDR に対し約 60%，また XDR を部分評価したものに対しても約 50% 程度，シリアライズのコストを削減できるといえる．

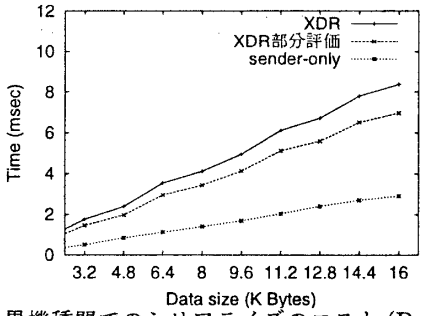

図 2: 異機種間でのシリアライズのコスト (Pentium II と UltraSPARC のマシン)

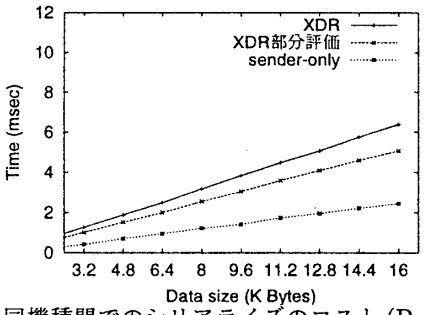

図 3: 同機種間でのシリアライズのコスト (Pentium II のマシン同士)

5 まとめ

RMI を最適化する一手法として，通信先の計算機のアーキテクチャを知り得た場合に，そのアーキテクチャで直接利用可能なオブジェクトを転送する方式を提案した．この方式を用いると，シリアライズのコストを従来の手法に比べ約 60% 削減できる．我々の研究グループで開発を進めている分散オブジェクトシステム [4] にこの手法を組み込む予定である．

参考文献

[1] Java Soft. *Java Remote Method Invocation Specification*, 1997. available from http://www.javasoft.com/.

[2] G. Muller, R. Marlet, C. Pu, and A. Goel. Fast, Optimized Sun RPC Using Automatic Program Specialization. In *Proceedings of IEEE 18th Int. Conf. on Distributed Computing Systems*, pages 240–249, 1998.

[3] Object Management Group. The Comon Object Request Broker: Architecture and specification, 2.0ed. Technical report, Object Management Group, July 1995.

[4] 河野健二, 加藤和彦, 益田隆司. 自律協調システムのための分散オブジェクトの共有機構. コンピュータソフトウェア (日本ソフトウェア科学会). 掲載予定.