2S−6

分散処理システムの相互接続のための RPCプロトコル変換

加藤 聰彦　藤長 昌彦
国際電信電話(株) 研究所

1. はじめに

複数の計算機を高速のネットワークで有機的に結合する分散処理システムの開発が広く行われている。現実の計算機環境では、異なった機種の計算機が目的に応じて使用されるため、分散処理システムについても、複数のシステムが共存しつつ普及していくと予想される。このような環境では、異なる分散処理システムを相互接続し、個々のシステムの特質を活かしながらより高度な分散処理を実現することが重要となる。

現在の分散処理システムの多くがクライアントサーバ・モデルに基づくRPC (Remote Procedure Call) に従っているため、分散処理システムの相互接続は、異なる分散処理システム上で稼働するクライアントとサーバの間でRPCを提供するという問題に帰着できる[1]。本稿では、分散処理システムの相互接続のためのRPCのプロトコル変換について、その課題と実現方法について考察する。

2. RPCによる分散処理システムの相互接続

RPCは、プログラムの手続き呼出しをネットワークに拡張したものである。サーバは提供する手続きの種類と引数のデータ型(インタフェース仕様と呼ぶ)を公開し、クライアントはその手続きを呼び出す。RPCの通信はメッセージの組み立て/分解並びに送受信を行うRPCスタブと、データ転送を実現するトランスポート機能により実現される。RPCスタブはサーバのインタフェース仕様からスタブ・ジェネレータにより自動的に生成される。

サーバは起動されると自分自身の論理名と通信に使用する識別子をネームサーバに登録し、クライアントはサーバを呼び出すために、その論理名を用いてネームサーバに識別子を問い合わせる。

RPCを用いた分散処理システムを相互接続するためには、図1に示すように、異なる分散処理システムに属する計算機ノード上のクライアントとサーバでRPCを可能とする必要がある。そのためには、RPCのプロトコル変換、トランスポート機能のプロトコル変換、異なる分散処理システム間の名前管理を実現する必要がある。以下では、分散処理システムの相互接続における基本的機能であるRPCプロトコル変換について考察する。

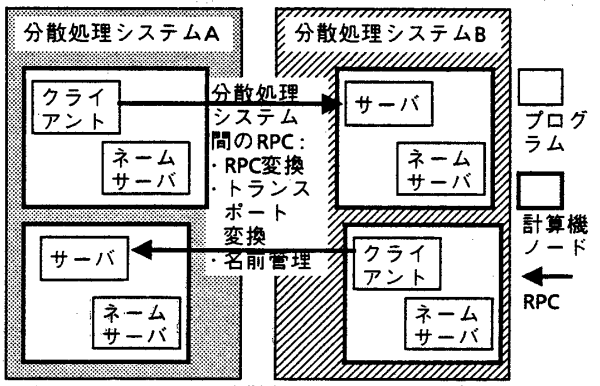

図1 RPCによる分散処理システムの相互接続

3. RPCのプロトコル変換の課題

(1) メッセージ形式の変換

分散処理システムでは固有のメッセージ形式を使用するため、その変換を行う必要がある。プログラマはRPCを用いて自由にサーバを開発するため、変換すべきインタフェース仕様は多数存在する。またメッセージを解析しただけでは、それが運ぶ引数のデータ型を判定できないのが一般的である。従って、RPCのメッセージ形式の変換においては、メッセージとサーバのインタフェース仕様の対応付けを行う必要がある。

(2) セマンティクスの整合

RPCには、それが実行された場合の作用が定められており、これをコール・セマンティクスと呼んでいる[2]。具体的には、at-most-once (RPCが成功した場合には、ただ1つのメッセージが転送されている)、at-least-once (同一のメッセージが2回以上転送される場合がある)、maybe (メッセージが転送されたかどうか不明である)などのコール・セマンティクスが考えられている。相互接続する分散処理システムにおいて、RPCのコール・セマンティクスが異なっている場合は、一方に合わせる必要がある。

またRPCが認証等の付加的な機能を持つ場合がある。付加機能を持たないRPCを用いる分散処理システムとその機能を持つ分散処理システムを相互接続する場合、不足の機能を補う必要がある。

(3) サーバのインタフェース仕様記述の変換

サーバのインタフェース仕様は、その分散処理システムのインタフェース仕様記述言語で表現され

RPC Protocol Conversion for Interconnection of Distributed Systems
Toshihiko KATO and Masahiko FUJINAGA
KDD R & D Laboratories

る。従って、他の分散処理システムのクライアントからRPCを行う場合は、インタフェース仕様記述の変換を行い、スタブを生成する必要がある。

4. RPCプロトコル変換の実現方法

4.1 RPC実行時の変換

RPCの実行時に、メッセージ形式とセマンティクスを変換する方法には次の3種類が考えられる。

① スタブにその分散処理システムの固有の通信を行う機能と相互接続用の通信機能の双方を持たせ、それを切り替える方法。

② サーバ毎にその専用のRPCゲートウェイを双方の分散処理システムに設ける方法。

③ すべてのサーバに対するアクセスを扱う汎用のRPCゲートウェイを設ける方法。この方法では、サーバのインタフェース仕様を管理するインタフェース・サーバを設ける必要がある。

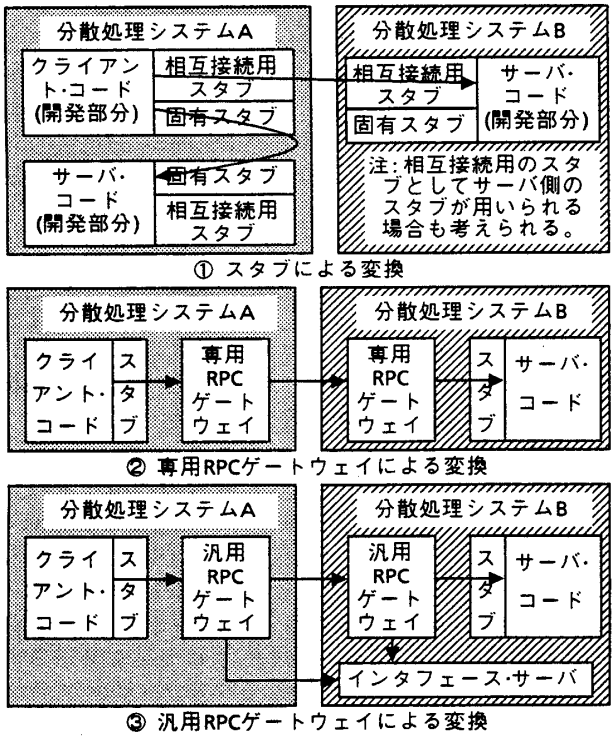

図2 RPCのプロトコル変換の方法

以下で、これらの方法を比較する。

(1) 利便性

スタブによる変換では、他の分散処理システムにアクセスするクライアントとサーバのスタブを変更し、再コンパイルする必要性があるため、既存サーバの活用という点で、利便性に問題がある。専用ゲートウェイによる方法は、容易に導入できるという利点はあるが、利便性をより高めるには専用ゲートウェイの生成等に対するサポート・ツールが必要である。3つの方法では、汎用ゲートウェイによる方法が最も利便性が高いと考えられる。

(2) メッセージ形式の変換への適用性

メッセージ形式の変換については、いずれの方法でも実現可能である。どの方式を使用するかは、利便性等他の要因を考慮して決定される。

(3) セマンティクスの整合の実現

スタブによる方法では、サーバ側の機能を持つスタブを相互接続に使用することにより、RPCのセマンティクスの整合をとることができる。

一方、ゲートウェイを用いてセマンティクスの整合をとるためには、サーバの仕様に別のパラメータを追加する等のより上位の機能が必要となる。例えば、at-least-onceのコール・セマンティクスのRPCを前提としている分散処理システムAのクライアントから、at-most-onceの分散処理システムBのサーバにアスセスする場合は、分散処理システムAにおいて同一のメッセージが2つ発生する可能性がある。これを防ぐためには、分散処理システムAにおけるそのサーバの手続きに、メッセージを識別するためのパラメータを追加し、ゲートウェイにおいて同一のメッセージが到着した場合は破棄する方法を用いる必要がある。このような処理は、サーバ毎に行う必要があるため、専用RPCゲートウェイを用いるのが適当である。

4.2 サーバのインタフェース仕様記述の変換

サーバのインタフェース仕様を、異なる分散処理システム上のクライアントに公開するためには、

① サーバ側のインタフェース記述言語で書かれた仕様を、クライアント側で使用する。

② 標準的なインタフェース記述言語を導入し、それでインタフェースを記述し公開する。

③ クライアント側のインタフェース記述言語でサーバのインタフェースを記述し使用する。

の3種類が考えられる。最も現実的な手法は第3の方法である。どの方法を用いるかについては、RPC実行時の変換方法や、標準RPCプロトコルの使用の有無、インタフェース記述言語の記述能力の違い等を考慮して定める必要がある。

5. むすび

本稿では、異なる分散処理システムを相互接続するためのRPCのプロトコル変換について、その課題を明らかにし、変換の実現方法に関して考察した。今後は、相互接続のための分散処理システムのモデルや、相互接続のプロトコル手順などについて検討を進める予定である。最後に日頃ご指導いただくKDD研究所小野所長、浦野次長、鈴木OSI通信グループリーダに感謝する。

参考文献

[1]: 藤長他, "RPCに基づく分散処理システムの相互接続に関する検討," 情報処理学会「1990年代の分散処理」シンポジウム, Nov. 1990.

[2]: A. Spector, "Perform Remote Operations Efficiently on a Local Computer Network," CACM, Vol. 25, No. 4, Apr. 1982.