7Q−1 異機種間リモートプロシジャコールシステム“KoKo”の設計

藤長昌彦　杉山敬三　加藤聰彦

国際電信電話株式会社 上福岡研究所

1. はじめに

複数の計算機を高速ネットワークで結合して機能分散/負荷分散を行ない、計算機資源の有効利用と性能向上を実現する分散処理技術への期待が高まっている。特に、クライアントサーバモデルに基づくリモートプロシジャコール(RPC)は、ネットワーク上の資源に関数呼び出しの形式でアクセスできるため、分散アプリケーションを開発する上で有効な技法である。このため、RPCを提供する分散オペレーティングシステム(OS)の研究/実用化が盛んに行なわれている[1,2]。

しかし、実際のネットワーク環境では種々のハードウェアやOSが混在して使用されることが多く、ひとつの分散OSに統一することは困難である。そこで、OSやハードウェアの異機種性に伴う通信機能の差異やデータ表現の違いを隠蔽する高速な異機種間RPCシステムを構築できれば、現実のネットワーク環境における分散アプリケーションの開発を容易にすることができる。

本稿では、UNIX、VMS、Machが混在するネットワーク環境を対象とし、筆者らが提案する“適応的データ表現変換”に基づいた、“KoKo”と呼ばれる異機種間RPCシステムの設計について述べる。

2. システムの概略

KoKoは、図1に示すように、

- スタブ
- 異機種間通信ライブラリ(HCLib: Heterogeneous Communication Library)、
- ネームサーバ(KNS: KoKo Name Server)
- スタブジェネレータ(KiG: KoKo Interface Generator)

から構成される。

KoKoにおけるサーバおよびクライアントの開発は、(1) サーバが提供する機能(RPCインタフェース仕様)の定義、(2) サーバ機能(図中“Server”の部分)の実現、(3) クライアント機能(“Client”の部分)の実現の手順で行なう。プログラマは、ネットワーク通信やデータ表現を意識することなく、クライアントとサーバが同一プログラム内に存在すると考えて、サービス固有の部分を開発するだけでよい。

サーバプロセスは、起動時にサーバ名と識別子をKNSに登録し、クライアントからの処理要求を待つ。クライアントプロセスは、サーバ名をキーにサーバの識別子をKNSに問い合わせて“クライアントとサーバの結合”を行ない、サーバ識別子を用いてRPCを実行する。

3. 設計方針

(1) 既存OSへの改修を行なわない: KoKoでは、既存OSの特質を活かした分散アプリケーションの開発を支援することが目的であるため、既存OSへの改修は行なわない。

(2) 適応的データ表現変換に基づくRPC: KoKoでは、通信を行なうふたつの計算機でデータの内部表現が異なる場合にのみデータ表現の変換を行なう、適応的データ表現変換[3]を採用することでネットワークRPCの高速化を図る。計算機内部で使用されるデータの表現形式は、バイトオーダ(複数バイトからなるデータの上位バイトが低いアドレスにおかれるか、高いアドレスにおかれるか)、アライメント(データをどのアドレスにも配置できるか、偶数アドレスに配置されるか、4バイトあるいは8バイト境界におかれるか)、実数の表現形式(IEEE-754フォーマットを用いるか、VAXフォーマットを用いるか)の3つのパラメータで実用上表すことができる。したがって、内部表現の形式には計算機の機種ほどの多様性はなく、データ表現の変換をまったく行なわずに通信できる可能性が潜在的に高いと考えられる。

クライアントとサーバが異なる計算機上で稼働する場合、両者の結合時にKNS間でデータの内部表現形式を交換してネゴシエーションを行ない、内部表現が異なる場合には計算機の処理能力と負荷を考慮して通信に用いるデータ表現を決定する。交換するパラメータが少ないため、ネゴシエーションのオーバヘッドはサーバの検索に比較して無視できる。

複雑なデータ構造を持つRPC引数においても表現変換を行なえるよう、変換はスタブで実行する。

(3) スタブの自動生成: スタブは、サーバのRPCインタフェース仕様からKiGにより自動生成される。

(4) ライブラリによるネットワーク透過なIPCの提供: HCLibは、OSごとに異なる通信機構をメッセージの送信、受信およびRPC呼び出しの3つの機能に抽象化し、OS独立かつネットワーク透過な形で提供する。通信の形態としては、各通信主体に一意の識別子を付与し、これを指定してメッセージの転送を行なうものとする[4]。識別子は、計算機内でユニークな計算機内識別子と計算機のIPアドレスの組からなり、これにより計算機内通信とネットワーク通信を切り替える。

(5) ネームサーバによるフォワーディング: ネットワーク通信では、サーバ側のKNSがクライアントとサーバ間

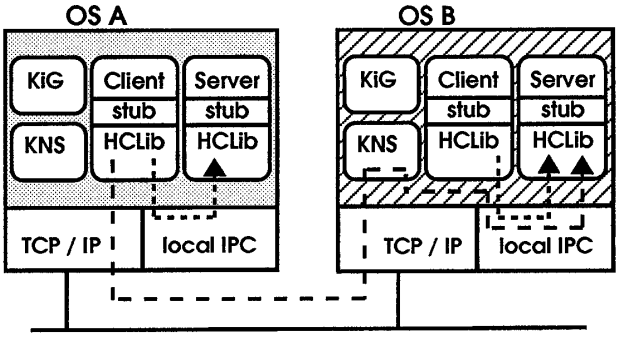

図1: KoKoの構成

“KoKo” : A Heterogeneous Remote Procedure Call System

Masahiko FUJINAGA, Keizo SUGIYAMA and Toshihiko KATO

KDD Kamifukuoka R&D Laboratories

の通信を仲介する。KNSによる転送はオーバヘッドを生じるが、不特定多数のクライアントからの非同期な処理要求を計算機内通信のみによって待つことができるため、HCLibの実現を単純化できる利点がある。

4. 詳細設計

4.1 KiGとスタブ

サーバのインタフェース仕様は、関数インタフェースを記述するファイルと、RPC引数の型定義を行なうファイルにより定義される。KiGはこのふたつのファイルを読み込み、サーバスタブ、クライアントスタブ、データ表現変換関数、外部関数宣言ファイルを生成する。この様子を図2に示す。C言語による開発環境を想定しているため、型定義ファイルおよび4つの出力ファイルはC言語で記述、生成される。ただし、C言語では関数引数の入出力関係が明確に記述できないため、関数インタフェースについてはKiG固有の宣言法をとった。

RPCの引数には、バイト、整数、実数の基本データ型およびそれらの配列、構造体を使用することができる。構造体のネスト、配列もサポートするが、ビットフィールド、'union'型、自己参照型構造体は使用できない。

KiGにより生成されるスタブは、データ表現の変換が不要な場合、RPC引数が複雑な構造体であっても、内部表現のままバッファにコピーしてRPCメッセージの組み立て/分解を行なう。表現変換が必要な場合には、構造体の要素をたどりながら、表現変換を実行しつつメッセージの組み立て/分解を行なう。

4.2 HCLib

HCLibは、OSごとに作成する。計算機内通信は、個々のOSに最適化されたOS固有のローカルIPC機構を利用して実現し、ネットワーク通信については、広く利用されているTCP/IPを用いる。

UNIXでは、UNIXドメインのデータグラムソケットを用いて計算機内通信を実現した。通信主体の識別子を受信用ソケットのソケット名と対応させている。VMSにおいてはmailboxを用い、受信用mailboxの“論理名”を識別子に対応させて実現した。

また、大量データの転送機能を“ポインタによるデータの受け渡し”のセマンティクスにより提供することとした。共有メモリの機能が利用できる場合には、これを用いてポインタデータの送受を行なう。SunOSではsystem Vの機能である共有メモリを、VMSではグローバルページセクションを利用して実現している。

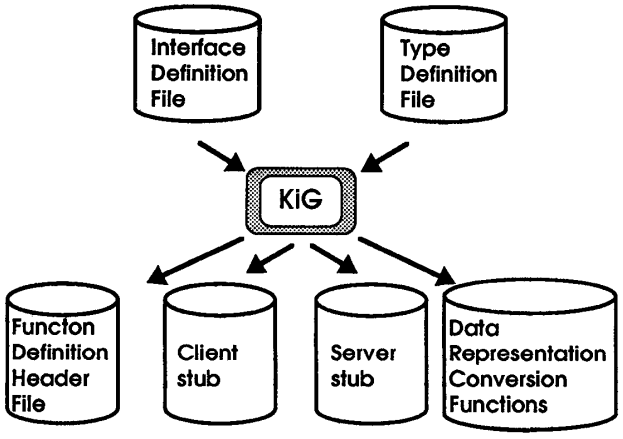

図2: KiGの動作

4.3 KNS

通信に用いるデータ表現は、クライアントへはサーバとの結合時にサーバの識別子と共に返し、サーバへは最初のRPCの直前に制御メッセージの形で通知する。

ネットワーク通信におけるメッセージ転送の機能については、UNIXでは'select()'システムコールを用いて、VMSではイベントフラグに対する'SYS$WFLOR()'システムコールを使って実現している。

5. 考察

(1) 異機種間RPCシステムの構築例としては、ワシントン大学のHCSプロジェクトがある[5]。HCSでは、異機種環境における“既存サーバへのアクセス”を主眼点としており、各種既存RPCのエミュレーションを行なうことで実現している。筆者らは、既存RPCとの互換性よりも、高速性に重きをおいている。データ表現に関して通信相手の内部表現をエミュレートする点で、HCSと類似しているが、これは表現変換に伴うオーバヘッドを取り除くためであり、RPCの枠組は単一のものを使用している。

(2) ネットワーク通信に標準形式を用いた方が望ましい場合もある。ブロードキャストにおける1:多通信や、ビットフィールドのように内部表現が機種により大きく異なる場合などがその例である。適応的データ表現変換は、“通信主体のどちらの内部表現に合わせるか”ではなく、“通信にどの表現を使用するか”をネゴシエーションするため、データの標準形式を導入することによりこの問題を自然に解決できる設計となっている。

(3) コンパイル時のオプションの違いによって、データの内部表現が変化する可能性がある。VAXにおける倍精度実数の2種類のフォーマットがその例である。これは特殊なケースであり、実用上の問題は生じないと考えて設計を行なった。しかし、必要であればサーバがデータの内部表現をサーバ名とともにネームサーバに登録することで解決することができる。

6. おわりに

複数の既存OSを含む異機種環境において、通信機構とデータ表現の差異を隠蔽し、表現変換に伴うオーバヘッドを最小とする異機種間RPCシステムの設計について述べた。本システムは、現実のネットワーク環境での分散アプリケーションの開発に有用であると考えられる。今後、本システムの評価を進める予定である。

最後に日頃ご指導戴くKDD上福岡研究所小野所長、浦野次長、鈴木コンピュータ通信研究室長に感謝する。

文献

[1] Avadis Tevanian, Jr and Richard F. Rashid. Mach: A basis for future unix development. Technical Report CMU-CS-87-139, June 1987.

[2] David R. Cheriton. The V distributed system. *Communications of the ACM*, 31(3):314–333, March 1988.

[3] 加藤聰彦 藤長昌彦 杉山敬三. 適応的にデータ表現変換を行なう リモートプロシジャコールの検討. 情報処理学会第40回全国大会, (5G–2), 1990.

[4] 藤長昌彦 杉山敬三 加藤聰彦. 異機種分散システムのための IPC プログラミングインタフェースの検討. 電子情報通信学会春季全国大会, (D–136), 1990.

[5] Brian N. Bershad et al. A remote procedure call facility for interconnecting heterogeneous computer systems. *IEEE Trans. on Software Engineering*, SE-13(8):880–894, Aug. 1987.