遠隔手続き呼び出しに基づいた分散C言語について

7M−5

村上 岳生　加藤 和彦　益田 隆司

東京大学　理学部　情報科学科

1　はじめに

分散システムにおけるシステム記述言語として遠隔手続き呼び出し[1]の概念に基づいた方法が現在広く使われている。この方法では既存のプログラミング言語と、rpcgen[5]、MIG[2]などの新たに設計されたインターフェース記述言語とを組み合わせて使用する事により分散システムの記述を行なう。この従来の方法では、プログラミング言語で記述された情報をほとんど使っていないためにプログラム本体とは別に遠隔手続きのためのインターフェースを記述しなければならない。また、ポインタ型の引数を使用する場合に参照渡しができない、引数に関数ポインタが使えないという欠点がある。

本研究では分散システムの記述を非分散システムとできるだけ同様に行なう事を目的として既存の言語仕様に最小限の拡張を施し、言語処理系のプリプロセッサ部にスタブ生成機能を組み込んだ。対象の言語としてはシステム記述に現在広く使われているC言語を選んだ。遠隔手続き呼び出しに関する機能拡張を行なったC言語を分散C言語と呼ぶ。

以下本稿ではC言語に対する機能拡張と分散C言語の実現法について述べる。

2　C言語文法の拡張

遠隔手続き呼び出しをC言語で直接利用するために以下のような文法を追加した。

2.1　関数の定義、宣言

サーバプログラムでは下のような記法で遠隔手続き呼び出しで利用する関数を定義する。サイト識別子変数には実行時に関数を呼び出したクライアントのサイト識別子が代入されプログラム中で参照する事ができる。

型指定子　関数名(引数宣言)@サイト識別子変数
{　関数本体　}

例

```
struct complex add(struct complex x,y)@ caller-id
{  /* body of the function add */  }
```

クライアント側のプログラムでは利用する遠隔手続きの宣言を行なう。また宣言、関数呼び出しのいずれかにサーバ識別子を指定する。

(宣言)
remote　型指定子　関数名(引数型宣言)@サーバ識別子;
(関数呼び出し)
関数名(引数値)@サーバ識別子;

例

```
(宣言)
remote complex add(complex, complex)@server-id;

(関数呼び出し)
z = add(x,y)@server-id;
```

この宣言を行なった遠隔手続きはローカルな関数と全く同様にクライアントプログラム中で利用する事が出来る。引数には基本データ型、構造体型、ポインタ型など通常の関数と同じデータ型を使用することが出来る。

2.2　ポインタ型引数

手続き呼び出しの際にポインタ型の引数を渡したい場合、遠隔手続き呼び出しにおいてはクライアントプログラムとサーバプログラムがアドレス空間を共有していないのでポインタが意味をなさない。そこでこれに対処するために2種類のポインタ情報受渡し機構を設け、プログラマーにどちらかを明示的に指定させることにした。1つはポインタの指しているデータのクロージャ全体を呼び出し時にクライアントからサーバへコピーし、手続きの実行終了時に再びサーバからクライアントへコピーする方法である。もう1つはあらかじめポインタのみをサーバへ渡し、ポインタ参照毎にクライアントを逆に呼び出し、必要なデータをコピーする方法である。前者の機構を **copy-on-call**、後者の機構を **copy-on-reference** と呼ぶ。両者ともに参照渡しが実現できるので意味的には同一であるが、copy-on-callはデータ量が少ない場合に有効であり、copy-on-referenceは大きなデータ構造の中から少量のデータが選択的に参照される場合に有効である。仮引数宣言時に下の例のようにリモートポインタ型と宣言するとcopy-on-reference、何も指定しない場合はcopy-on-callの機構が用いられる。copy-on-referenceの場合、リモートポインタ型として使用する変数は明示的に宣言を行ない、リモートポインタ型とローカルポインタ型の間での代入は許されない。

```
void func(remote struct list *l)@caller-id
{
  remote struct list *temp;

  for (temp = l; temp!=NULL; temp=temp->next){
        if (temp->item == ...) ...
}
```

配列型データをcopy-on-callで遠隔手続きに渡す場合は要素数を明示的に指定する。クライアントプログラムでは関数宣言時に要素数の最大値を指定し、関数呼び出し時に実際に送る要素数を指定する。サーバプログラムではクライアントプログラムから実際に受け取る要素数を代入する変数を宣言する。

例

```
(サーバプログラム)
void sort([n]int *data)@caller-id
{  /* body of the function sort */  }

(クライアントプログラム)
remote void sort([100]int *);

int *data;
sort([40]data)@server-id;
```

3　分散C言語の実現

遠隔手続き呼び出しはスタブを使用する事によって実現される。スタブとはクライアント側、サーバ側にそれぞれ存在し、送信、受信メッセージのエンコード／デコード、クライアント／サーバ間の

On the Distributed C Language Based on Remote Procedure Calls
Takeo MURAKAMI, Kazuhiko KATO, and Takashi MASUDA
Dept. of Information Science, Univ. of Tokyo.

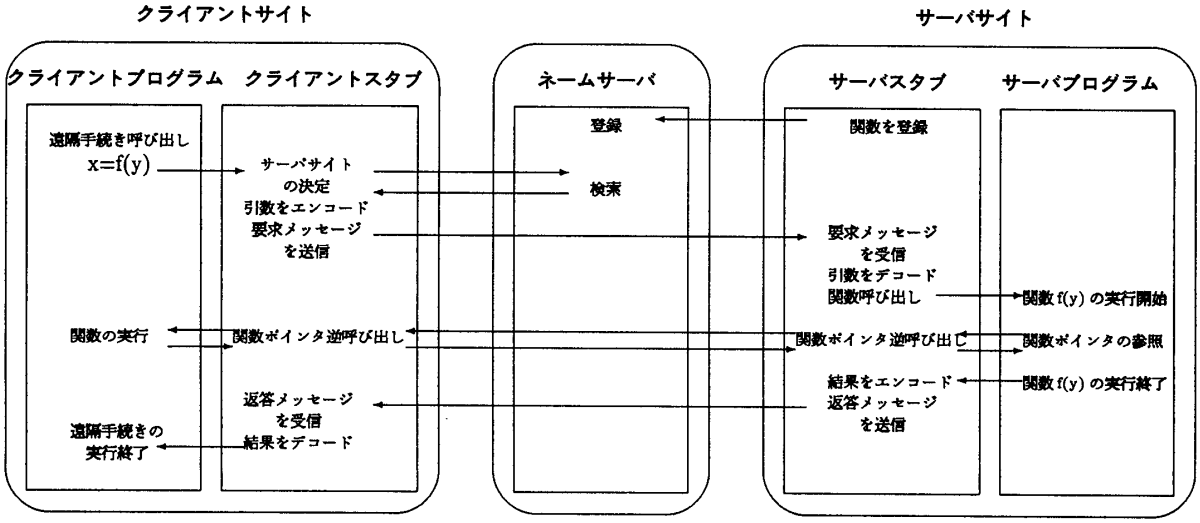

図 1: 遠隔手続き呼び出しの実行機構

通信の処理を行なうモジュールのことをいう [1]。図1は遠隔手続きが実行される際の処理の流れを表している。

3.1 クライアント / サーバ間の通信路の確立

遠隔手続き呼び出しを利用する際に通常の関数呼び出しと異なるのはサーバ識別子を指定する点である。サーバ識別子は遠隔手続きを提供するサーバのサイト名またはサイトグループ名である。サーバ識別子にサイト名を指定した場合はクライアントはサーバを直接呼び出す。サイトグループ名を指定した場合はネームサーバを使用してサーバのサイトを決定する。ネームサーバはサイトグループ名とサイト名、提供している手続き名の情報を保持している。サーバはサイトグループ名と提供する遠隔手続き名をあらかじめネームサーバに登録しておく。クライアントからの問い合わせがネームサーバに届くとネームサーバはサイトグループの中から適当なサイトを一つ選び、そのサイト名をクライアントへ返す。これらの処理はスタブによって行なわれるのでプログラマーからは透明である。

3.2 引数のエンコード / デコード

遠隔手続き呼び出しで使われる引数、結果の値は発送前に外部データ表現フォーマットに変換され、受信時に再び計算機の内部データ表現に変換される。外部データ表現フォーマットを用いる事により異なる内部データ表現を用いる機種間での遠隔手続き呼び出しを行なう事が可能となる。現在の実現では外部データ表現フォーマットとしてSun XDR [4] を利用している。

3.3 ポインタ型の引数

2章で述べたようにポインタ型の引数に対する実行機構は2種類ある。このうちcopy-on-reference機構の場合、リモートポインタ型とローカルポインタ型を明確に区別しているのは次のような理由による。この区別をせずに実現しようとした場合に問題となるのはリモートポインタ型の引数を他のポインタ型の変数に代入する場合である。この場合ポインタ参照がリモートなのかローカルなのかをコンパイル時に判断するのは困難である。これに対していくつかの解決策が考えられる。ひとつはこのような代入を許さない事である。しかしリンクしたリストをたどる場合などにはこの代入が必要であり、これではプログラマーへの制約がきつすぎる。またポインタにローカルかリモートかを識別するタグを付加し、実行時にタグを調べる事によってポインタ参照を行なう方法も考えられる。この場合はすべてのポインタ参照について実行時にチェックが必要で効率が悪い。以上のような考察からリモートポインタ型として使用する変数は明示的に宣言を行なう事にした。

ポインタを実行時にたどるのは**逆呼び出し** (back-call) と名付けた機構によって行なう。逆呼び出しとはサーバプログラムの実行中にクライアントを呼び出し、クライアントのアドレス空間に存在するデータや関数を参照する機構である。2.2で述べたcopy-on-reference機構と関数ポインタは逆呼び出しの機構を用いて実現される。図1の様にサーバプログラム中で関数ポインタが参照される毎にクライアントに存在する関数を呼び出す。この逆呼び出しのためのスタブが通常のスタブとは別に処理系によって自動生成される。サーバプログラム中での関数ポインタ参照はスタブの呼び出しに置き換えられる。スタブでは通常の遠隔手続き呼び出しの場合と同様な処理が行なわれ、プログラマーには透明な形でクライアントプログラム中の関数を呼び出すことが出来る。

3.4 処理系の構成

分散C言語の処理系はCコンパイラとプリプロセッサを組み合わせることにより実現されている。プリプロセッサは分散C言語のソースコードを読み込み、C言語のソースコードを出力する。このプリプロセッサはCOB[3] のプリプロセッサを一部書き換えることによって実現した。

4 おわりに

言語処理系の中にスタブ生成機能を組み込んだために、プログラミング言語によって記述された手続き宣言、および変数の型宣言に関する情報を利用する事が可能となった。このためプログラマーは遠隔手続きの仕様を特別に記述する必要がなく、遠隔手続きを普通の手続きと同様に扱う事が可能である。

謝辞

IBM東京基礎研究所の上村 務氏と小野寺民也氏にはCOBの使用に関して協力して頂きました。ここに感謝致します。

参考文献

[1] Birrell, A.D. and B.J.Nelson,"Implementing Remote Procedure Calls," *ACM Transactions on Computer Systems*, Vol.2, No.1, February 1984, pp.39-59.

[2] Draves, R.P., M.B.Jones, and M.R.Thompson, "MIG - the MACH interface generator," *Technical report*, CMU, February 1988.

[3] 上村, 横内, 吉田, 大平, J.M.Lucassen, "COBにおけるオブジェクト指向機能," コンピュータソフトウェア, Vol.6, No.1, January 1989, pp.4-16.

[4] *External Data Representation Standard: Protocol Specification*, Sun Microsystems Inc., 1988.

[5] *rpcgen Programming Guide*, Sun Microsystems Inc., 1988.