オブジェクト指向分散環境 OZ++の通信機構の実装

2T－1

濱崎 陽一 (電子技術総合研究所)、大西 雅夫* (東洋情報システム)、
鈴木 敬行* (シャープビジネスコンピュータ)、中村 章人 (電子技術総合研究所)、
塚本 享治 (電子技術総合研究所)

*: 情報処理振興事業協会「開放型基盤ソフトウェア研究開発評価事業」研究員

1　はじめに

オブジェクト指向分散環境OZ++では、オブジェクト間でメソッドの呼び出しにより実行が進み、そのときにオブジェクトが引数としてあるいは結果として交換される。オブジェクトは相互に参照関係があり、そうした相互の関係を保存したままでオブジェクトを交換することが出来るのがOZ++の特徴の一つである [2]。

通信機構には、メソッドの呼び出し、通信アドレスの解決、ブロードキャストなどの機能があるが、本稿ではメソッドの遠隔呼び出しに関わる通信機構の実装を述べる。

2　OZ++システムの構成

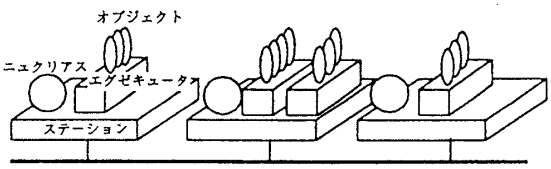

図 1: OZ++ システムの構成

OZ++ システムの構成を図1に示す。OZ++は、ネットワークで接続された複数のステーション (計算機) 上に実装される。

エグゼキュータは、オブジェクトを実装し、そのメソッドのマルチスレッドによる実行やオブジェクト間の通信を行う [4]。各エグゼキュータは通信アドレスを持っており、相手側のアドレスを指定して通信を行う。

ニュクリアスは、エグゼキュータを管理するもので、エグゼキュータの通信アドレス解決の機能を持っている。

An Implementation of the communication mechanisms of OZ++ : An Object-Oriented Distributed Processing Environment
Yoichi Hamazaki(Electrotechnical Laboratory),
Masao Onishi*(Toyo Information Systems, Co., Ltd.),
Takayuki Suzuki*(Sharp Business Computer Software, Co., Ltd.),
Akihito Nakamura(Electrotechnical Laboratory),
and Michiharu Tsukamoto(Electrotechnical Laboratory)
*: Research fellow of Open Fundamental Software Technology Project in Information-technology Promotion Agency, Japan

3　OZ++のオブジェクト

OZ++のオブジェクトにはシステム全体でユニークな名前 (OID) を持っていてどこからでもアクセス可能なグローバルオブジェクトと、グローバルオブジェクトの部品として使われるローカルオブジェクトがある [3]。ローカルオブジェクトはポインタにより参照され、グローバルオブジェクトはOIDにより参照される。ローカルオブジェクトをグローバルオブジェクト間で交換する場合には値渡しとなり、参照されているローカルオブジェクトと共に転送される (図 2)。

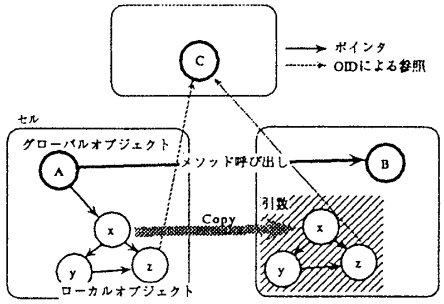

図 2: グローバル / ローカルオブジェクト

エグゼキュータはシステム全体でユニークな名前 (ID) を持ち、OIDはそれにエグゼキュータ独自の番号を付け加えた構造になっている。オブジェクトが存在するエグゼキュータの通信アドレスは、エグゼキュータのIDから求め得るようにニュクリアスにより管理されている。

4　通信機構の実装

エグゼキュータ間のメソッドの呼び出しは次のように実装されている。

4.1　チャネル

グローバルオブジェクトのメソッドが呼び出された時には、新たにスレッド (実行スレッド) が生成され、その実行スレッドによりメソッドが実行される。メソッド呼び出しに必要な情報の交換、スレッド間の同期ために、チャネルと呼ぶ構造を用いた。チャネルは呼び出し側と呼び出され

側に分かれておりそれぞれ送信チャネル、受信チャネルと呼ぶ。メソッドを受け取るオブジェクトの位置が同一エグゼキュータ上にあるか否かによりスレッドの生成や引数などの受渡しの方法が異なるので、その違いを吸収するために抽象化がされている。以下、異なるエグゼキュータ間のチャネルについて述べる。

4.2 チャネル間通信

グローバルオブジェクトのメソッド呼び出しの様子を図3に示す。送信チャネルから呼び出されるオブジェクトのあるエグゼキュータにCALL_INDICATIONパケットによりOIDとメソッドのセレクタが送信される。CALL_INDICATIONの受信により受信チャネルが生成される。次に、引数がCALL_ARGSパケットにより受信チャネルに送られ、引数が揃ったところで実行スレッドが生成されてメソッドの実行が始まる。メソッド実行の結果はRESULTパケットにより送信チャネルに送られ、呼び出したスレッドに渡される。

一般のRPCと異なり、呼び出しを2段階に分けた。これは、受信チャネルの生成とそれに先立つオブジェクトの検索などの処理と、引数のエンコードの処理を並行して行えるようにするためである。

CALL_ARGSとRESULTパケットにはエンコードされたオブジェクトが含まれ、可変長のパケットとなる。また、複数のパケットに分割して送られる場合がある。分割して送られる場合も、1パスのエンコーダを用いているためにエンコードの処理と送信とを並行して行う事ができる。

チャネル間の通信はTCP/IPを用いて実装した。コネクションはエグゼキュータ間で張り、複数のメソッド呼び出しが同じコネクションを使う。また、エグゼキュータは制限数以上のコネクションが必要な時には、あまり使われていないコネクションを切ってそれを再利用する。

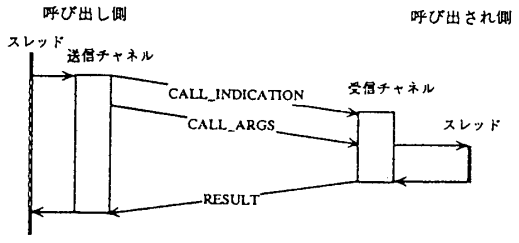

図 3: チャネル間の通信

4.3 エンコード / デコード

エンコーダには、オブジェクトのメモリイメージに近い形でエンコードするLight Weight Encodingの手法を用いた。ローカルオブジェクトへの参照はポインタであるので、これらをパケット内で意味のあるローカルなインデックスにエンコードしなければならない。また、ローカルオブジェクトは、そのローカルオブジェから直接 / 間接に参照されている全てのローカルオブジェクトを含めて、その参照関係を保ったままエンコードされる必要が有る。そこで、FIFOと重複エンコードを避けるためのテーブル(TBL)とを用いた1パスのエンコードを開発した。エンコードのアルゴリズムの概略は(1)FIFOに最初のローカルオブジェクトのアドレスを入れる(2)FIFOからアドレスを取り出す。FIFOが空なら終り。(3)そのオブジェクトをポインタ以外はメモリイメージのままパケットに詰める。オブジェクトを参照しているポインタがあれば、それをTBLから探して、なければFIFOに入れるとともに新しいインデックスを与えてTBLに記録する。ポインタはTBLに記録されたインデックスに変換する。(4)(2)から繰り返す。デコードはエンコードと逆の処理をするが、インデックスからポインタへの変換は全てのオブジェクトをメモリ上に展開した後におこなうので2パスとなる。

5 性能測定

下表は、1整数を引数として整数を返すメソッドの呼び出しにかかる時間を同一ステーション上のエグゼキュータ間(SS2上、SS10上)、および異なるステーション間の場合で測定したものである。異なるステーション間のほうが、処理が並行に行われるため、ネットワークによる通信が含まれるにもかかわらず若干処理が速い。

SS2 上	SS10 上	SS2 間	SS10 間
13.7ms	7.6ms	10.2ms	6.7ms

6 まとめ

OZ++の通信機構の実装について、遠隔オブジェクトのメソッド呼び出しにかかわる機能を中心に述べた。現在、OZ++システムの第1版の実装を終え、評価中である。今後、異機種間通信への拡張を計画しておりその際にはエンコード / デコードにおいて機種依存データの変換機能を付け加える必要が有る。

本研究は、情報処理振興事業協会「開放型基盤ソフトウェア研究開発評価事業」の一環として行われたものである。

参考文献

[1] 浜崎他:「オブジェクト指向分散環境OZ++の通信機構の基本設計」、情報処理学会第46回全国大会、Mar.1993

[2] 塚本他:「オブジェクト指向分散環境OZ++の基本設計」、SWoPP'93、Aug.1993

[3] 新部他:「オブジェクト指向分散環境OZ++の実行モデル」、情報処理学会第47回全国大会、Oct.1993

[4] 浜崎他:オブジェクト指向分散環境OZ++の実行機構の設計」、情報処理学会第48回全国大会、Mar.1994