Vol. 36 No. 8　　情　報　処　理　　Aug. 1995

解　説

日本におけるオペレーティングシステム研究の動向

1. 4　オブジェクト指向開放型分散システム：OZ+†

塚本享治††　濱崎陽一††

1. はじめに

計算機のダウンサイジング，オープン化によりもたらされる異機種分散環境においては，プログラムが各所で開発され，それらのデータが複数のユーザにより共有，交換できることが期待されている．しかしながら，データを受信したエンドシステム（計算機）にそれを処理するプログラムが常に存在するとは限らず，その場合には，所望のプログラムの検索，入手，組込みに多大な労力が必要となる．また，開発されたプログラムを利用・改変して新たなプログラムを開発し，安定に動作するようにするのも難しい．このような問題を解決するために，オブジェクト指向開放型分散システムOZ+を開発した[6)~8)]．

OZ+では，オブジェクト指向の概念に基づいてすべての情報をオブジェクトとして表現する．オブジェクトの仕様（クラス）を分散システム全体で系統的に管理することによりクラスの共有が可能となる．また，分散システム上でクラスを継承することにより，既存のクラスの利用・改変が容易であるというオブジェクト指向の利点が分散環境においても享受できる．

オブジェクトをネットワーク上で転送可能とし，その際，オブジェクト間の参照関係を維持し，しかも，オブジェクトが転送された先に必要なクラスが存在しないときには，クラスの管理機構から取得する．

このような機能を実現するために，ネットワーク上のオブジェクトを記述する分散オブジェクト指向型言語，オブジェクトを交換しながら計算処理を進める分散計算システム，オブジェクト群を管理する分散管理システムを開発し，これらを統合したものがOZ+システムである．

2. 分散オブジェクト指向型言語

分散システム向きの言語は，分散されたメモリへのアクセスが抽象化され局所化できる必要がある．また，システムの拡張が差分的に行えるようにするために，継承機能を持つオブジェクト指向言語とすることにした．

その設計にあたっては，Smalltalk 80[1)]に倣って単一継承のタイプレス言語としたが，分散システムのために，次のような工夫を行った．

（1）　オブジェクトの相互作用

オブジェクトを大域的と局所的の2種類に分けることとした（類似のものはArugus[2)]）．OZ+ではメソッドを呼ぶ際の引数と結果はオブジェクトであり，異なるエンドシステム上のオブジェクトのメソッドを呼ぶ際に引数を呼ぶ側または結果を返す側に置いておくと，引数や結果を受け取った側からの呼び返しの際に性能が低下するからである．

大域的オブジェクトとは分散システム上で一意な識別子（UID；Unique ID）で識別されるオブジェクトであり，局所的オブジェクトとはいずれか1つの大域的オブジェクトの部品となる，UIDを持たないオブジェクトである．大域的オブジェクトまたはそれに属する局所的オブジェクト間でメソッドを呼ぶ際の引数や結果に現われるオブジェクトはポインタ渡し（call-by-reference）とするが，他の大域的オブジェクトを呼ぶ際の引数または結果に現われる局所的オブジェクト群はコピーし（call-by-value），大域的オブジェクトはUIDだけを渡す方式とした．**図-1**にオブジェクト交換の例を示す．大域的オブジェクトAから大域的オブジェクトFのメソッドを局所的オブ

† Object-Oriented Open Distributed System：OZ+ by Michiharu TSUKAMOTO and Yoichi HAMAZAKI (Electrotechnical Laboratory).

†† 電子技術総合研究所

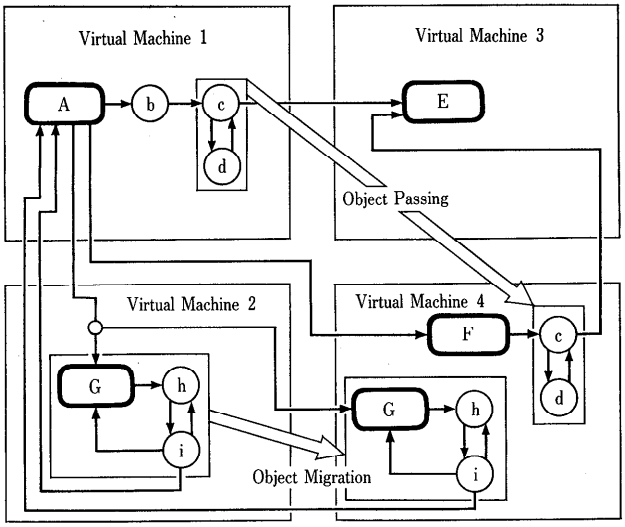


図-1　オブジェクトの転送

ジェクトcを引数として呼び出すとき，cのみならずcから参照されている局所的オブジェクトdもコピーして渡す．オブジェクトcとdの相互参照関係や大域的オブジェクトEへの参照はコピーにおいても保存される．

（2）　オブジェクトモデル

オブジェクトを能動的なものと受動的なものに分けた[3]．Smalltalk 80ではオブジェクトは受動的であり，実行を行うプロセスはメソッド呼び出しの連鎖に沿ってオブジェクトを次々に渡り歩く．この方式は，単一計算機上では効率よく動作するが，分散したオブジェクトの場合にはエンドシステムにまたがるため，その効率的な実現は難しいからである．

能動的オブジェクトのメソッドが呼ばれた際には，呼ぶ側のプロセスは保留状態になり，呼ばれた側にプロセスを作って実行され，結果が戻されると保留中のプロセスが再開される．受動的オブジェクトのメソッドが呼ばれた際には，呼ぶ側のプロセスで実行される．メソッドの呼び出しがどこから行われるかを考えれば能動的オブジェクトとすべきオブジェクトは他のエンドシステム上のオブジェクトからアクセスされうる大域的オブジェクトであり，受動的オブジェクトとすべきオブジェクトは必ず同じエンドシステム上に存在する局所的オブジェクトであることは明白である．つまりOZ＋のオブジェクトは大域的で能動的なオブジェクトと局所的で受動的なオブジェクトに分類される．

大域的オブジェクトのメソッドの呼び出しには，結果の受信を待つ同期型コール，すぐに次の実行に移って，結果が入る場所に対するデータ要求の際に同期をとる非同期型コール，結果を受信しない非同期分岐の3種類がある．

大域的オブジェクトはモニタ[4]になっており，その内部で同時に実行できるプロセスは1つに限られる．実行中のプロセスが条件待ちになるか，メソッドの実行を終了すると，実行可能な条件にある，あるいはメソッドの入り口で待つ他のプロセスが実行される．メソッドの実行の途中で他の大域的オブジェクトのメソッドを同期型コールで呼び出す際に呼び出し側の大域的オブジェクトをロックしたままにすると呼び戻しが起った際などにデッドロックに陥いる．そこで他の大域的オブジェクトのメソッドを呼び出す際にはロックを解放する方法[5]を用いた．

（3）　オブジェクトのグループ化

同種のオブジェクトをグループ化し，グループメンバに対して同じメソッド呼び出しを行う機能を導入した．分散システムの管理システムでは，各エンドシステムに同種の機能を持ったエージェントを配置し，それらに対していっせいに問合せや操作を行うことが多いからである．

単一のオブジェクトに対する呼び出しとグループに対する呼び出しを統一的に扱えるように前述の3種類の呼び出しを，1つの結果を受信すると

次に進む単一値型コール，結果の入る配列へのポインタが返されてすぐ次に進み，その配列にアクセスする際に同期をとる配列値型非同期コール，結果を待たない非同期分岐にそれぞれ拡張した．

3. 分散計算システム

分散オブジェクト指向型言語で記述されたオブジェクトを稼働させる分散計算システムは，エンドシステムを接続するネットワーク，ネイティブなOSを拡張した分散カーネル，オブジェクトを搭載して実行する仮想計算機から構成される．

異機種間でクラスコードを共有し，移動可能なオブジェクトを実現するために，仮想計算機は機種に依存しない実行環境を実現するものとし，分散オブジェクト指向型言語で書かれたプログラムをコンパイルして得られるクラスコードを解釈するバイトコードインタプリタとして実現した．各エンドシステムには分散カーネルと任意個の仮想計算機が存在する．

3.1 分散カーネル

分散カーネルはローカルなネイティブOSと統合して仮想計算機を実装するのに必要最低限の分散OS機能を提供する．分散カーネルに組み込んで実現した主な機能には次のようなものがある．

（1） データ転送機能

各種ネットワークで多数の移動可能なオブジェクト間で効率の良い通信と管理を実現する同報機能を有するコネクションレス型データ転送プロトコルである．このプロトコルはオブジェクトのマイグレーション時にアドレスの付け替え，仮想計算機の動的なアドレス管理，中継機能などを持つ．

（2） グループ同報RPC機能

複数のオブジェクトへの呼び出しを単一のオブジェクトへの呼び出しと同様に行うことのできるグループ同報機能を持つRPCプロトコルである．グループへの加入や離脱時の一貫性管理はディレクトリ方式のキャッシュによって実現した．

3.2 仮想計算機

仮想計算機は，オブジェクトを搭載し，相互にオブジェクトを交換しながら，オブジェクトの実行を行う．主な機能とその実装の概略は次のとおりである．

（1） オブジェクトのデータモデル

オブジェクトはプロクシとオブジェクトの実体を格納するセルに分けて構成した．ここでいうプロクシとはセルがなくてもそのオブジェクトの属性が分かるようにするための種々のタグ，セルのメモリアドレス，オブジェクトのマイグレーション先アドレスなどを持つもので，表形式で管理している．大域的オブジェクトへの参照はUIDとアドレスを持つプロクシにより実現した．

（2） オブジェクト転送機能

他の大域的オブジェクトのメソッドが呼ばれる際の引数が局所的オブジェクトの場合には，それが参照している局所的オブジェクトも一緒にコピーして送られる．オブジェクトの符号化の際には，局所オブジェクトを識別しパケット内でポインタの役目をするために，連番（LID: Local ID）を振った．大域的オブジェクトはUIDを含むプロクシのみを，局所的オブジェクトはメモリアドレスをLIDに変換したプロクシおよびセルを再帰的に符号化することにより直接または間接的に参照されているオブジェクト群が符号化される．

さらに，大城的オブジェクトで実行中のプロセスの中断，大域的オブジェクトの移動，元のオブジェクトの削除，移動先でのプロセスの再開を順次行うことにより，オブジェクトのマイグレーションも可能となった（図-1）．マイグレーションしたときには，移動先の情報を持ったプロクシを元の仮想計算機に残しておき，マイグレーションしたオブジェクトに要求が届いた際に移動先を教えることにより，要求元でプロクシを書き換え，移動先へ要求しなおすようになっている．

（3） グループ管理機能

オブジェクトのグループに対する呼び出しを実現するためにグループにもUIDを与えたが，オブジェクトのグループ自体はオブジェクトではない．オブジェクトのグループへの参照はグループのUIDを持つプロクシにより行い，グループのUIDとグループのメンバの管理およびグループのメンバへのメッセージの配送は分散カーネルにより実現した．

（4） クラスロード機能

オブジェクトの転送により転送されるのはインスタンスのみであり，クラスコードは転送されない．これはクラスコードは一般に長大であり，す

でに受信側にロードされている場合が多いと予想されたためである．そのためにクラスコードがロードされていない際には何等かの方法でロードする機構が必要である．OZ+ではクラスのUIDからそのクラスを管理するクラスサーバのUIDが認識できるように階層的にUIDが振られている．クラスがロードされていなければそれを管理するクラスサーバに要求を出す．あるクラスに関する要求に対して，そのクラスおよびすべての上位クラスのうち未ロードであるクラスコード群が返されるので，実行に際して発生するクラスロードはたかだか一回である．

4. 分散管理システム

複数利用者を対象としたシステムとするためには，システム全体でオブジェクトを管理し，また各利用者ごとの環境を維持管理する必要がある．前者をオブジェクト管理と呼び，後者をアカウント管理と呼ぶ．

分散管理システムは分散オブジェクト指向型言語で開発し，オブジェクトによって実現した．管理機能を仮想計算機ではなくオブジェクトにより実現したことで，管理システムの拡張や改変が容易になっている．

4.1 オブジェクト管理

複数利用者間でオブジェクトの共有と交換を行うには，オブジェクトの名称と実体の対応付け，名称の共有，クラスの管理，オブジェクトの生成者よりも寿命の長いオブジェクトの実現が不可欠である．

（1） 名称とディレクトリ

大域的オブジェクトはUIDで識別されるが，利用者には理解しにくい．そこで，文字列の名称とUIDの対応付けを行うネームサーバを導入した．ネームサーバは名称空間を構成するものであり，サーバを分散化することで効率化を図った．また，名称やUIDにより識別されるオブジェクトへのアクセスを可能にするために，そのオブジェクトの実体に写像するディレクトリサーバも導入した．

（2） クラス管理

単一継承を採用したので，すべてのクラスは1つのクラスを最上位クラスとする木構造（継承木）をなす．クラス群の管理を容易にするために，木を分割して管理する．継承木の一部をなすクラス群を記憶，管理するのがクラスサーバである．クラスサーバはUIDによる問合せに対してクラスコード（オブジェクト）を戻す．クラスサーバはクラスごとにどの仮想計算機にクラスを供給したかを記録している．あるクラスに関する問合せに対してそのクラスとその上位クラスの供給記録を調べ，未供給であるクラスコードすべてをまとめて供給する．上位クラスが別のクラスサーバに管理されている場合にはそのクラスサーバからクラスコードを取得する．**図-2**にクラスロードの例を示す．その名称がVMID1である仮想計算機からClassEに関する問合せを受けたClassServer2は，ClassEとその親クラスClass-Cを調べ，ClassServer1にClassBを問い合わせる．ClassServer1はすでに供給しているObjectクラスを除いてClassBのクラスコードを返し，ClassServer2はそれにClassCとClassEのクラスコードを併せて仮想計算機VMID1に返す．

クラスは別の利用者が利用している可能性があり，クラスの改変や消去が問題を引き起こす可能性がある．そこで，クラスの記述にバージョン番号を指定することにより解決を図った．また，バージョンに留意せずとも利用できるように，公開バージョンという概念を導入し，バージョン指定を省略した場合にもUIDが決定できるようにした．

（3） 永続オブジェクト

利用者間でオブジェクトを交換する際，送信者と受信者が同時にシステム内に存在することを要求するのは現実的でない．そこで，送信したオブジェクトが二次記憶に格納され，ある有限時間後に受信者により取り出されても正しく動作するように，つまり，送信者と受信者の生存期間とは独立な生存期間を持つ永続オブジェクトを導入した．

オブジェクトの永続化に必要な条件は，二次記憶に格納されるオブジェクトが参照するオブジェクトがすべて二次記憶上に存在することである．このために，システムに永続性を保証するシステム空間を設け，ネームサーバとクラスサーバをシステム空間に置いた．さらに，利用者ごとに永続環境を記憶するアカウントエージェントをシステム空間に置いた．他の利用者にオブジェクトを送信するのに先だって，送信するオブジェクト群が

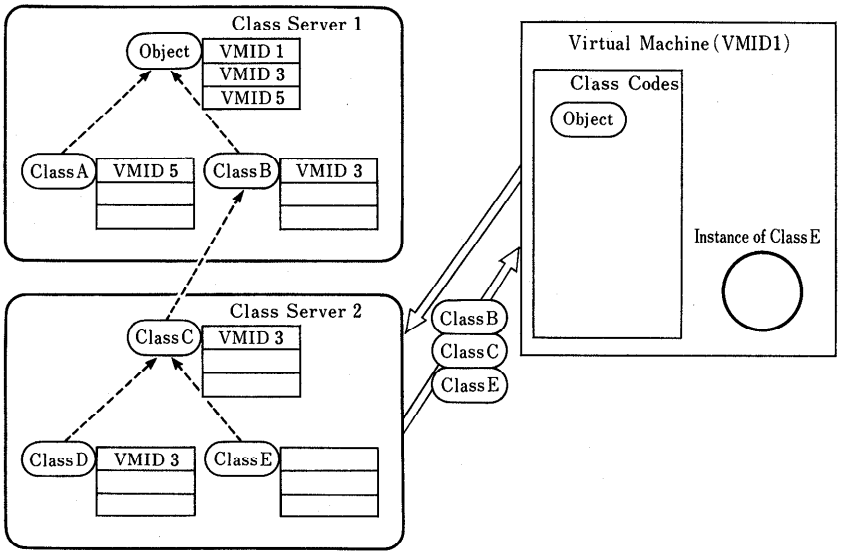


図-2 クラスのロード

参照しているために呼び返しが発生する可能性のあるオブジェクトを自分のアカウントエージェントに記憶する．これにより，発信者が消滅した後もアカウントエージェントが呼び返しを代行することができるので，問題が生じない．

4.2 アカウント管理

利用者ごとの環境を維持管理するアカウント管理は，次の二種類のオブジェクトにより実現した．

（1） アカウントエージェント

利用者ごとにアカウントエージェントは作成され，その利用者に必要な初期環境を作って二次記憶上に記憶する．対応する利用者がログインしていないときには，二次記憶から呼び出されてその利用者の代行処理を行う機能を持つ．

（2） ユーザエージェント

ログイン単位で利用者の一時的な環境を維持管理するのがユーザエージェントである．利用者がログインする際に動的に作成され，対応するアカウントエージェントから必要な環境をロードする．また，機能分散のために他のエンドシステムに作業用仮想計算機を作成することができ，作業用仮想計算機からの要求を受けて名称やクラスを配布する機能を持つ．

5. おわりに

OZ+は，(1)ネットワーク上でオブジェクト群を相互関係を維持したままで交換することができ，分散システムにおいても単一システムとまったく同じ状況で処理可能である，(2)受信側にクラスが存在しないときにも必要に応じてクラスを自動的にロードできるので発信側がクラスを変更・拡張しても受信側では正しく解釈実行できる，(3)継承管理とバージョン管理を行っているため分散したクラスの変更・拡張にも解釈・実行が正しくできる，という特徴がある．これにより複数利用者間で相互運用性が保証されたシステムとなった．

現在，OZ+をベースに機能の高度化と性能の改善を図るOZ++システムの開発[9),10)]を進めている．OZ+からOZ++に移行するにあたっては，次の点に大きな改良を加えた．

（1） 単一継承型言語から，多重継承型言語へ

クラスのモジュラリティを高め，継承によるクラスの再利用をより自然に行えるように，単一継承から多重継承が可能な言語とした．

（2） タイプレス言語からタイプ付言語へ

OZ+では実行時の例外処理によってプログラミングの誤りなどを解決する方法をとったが，デバッグなどが困難であった．そこで言語をタイプレスからタイプ付言語とし，コンパイルによって

タイプチェックをすることにより実行時に発生するエラーを事前に検出することとした．

（3） クラスのバージョン管理の高度化

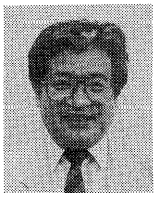
OZ＋ではクラスにバージョンを指定することができたが，バージョン付与の方針などについてはプログラマに任されていてシステムによるサポートが十分でなかった．ダイナミックロード時のリテラルへの名称付与で行っていたため実行時のオーバヘッドも大きかった．そこでクラスの利用の方法とそのときのインタフェースに着目したバージョン管理を導入し，バージョンの同一性の意味を明確にし，再コンパイルの必要性の判断も可能にした．

なお，OZ＋の研究は大型プロジェクト「電子計算機相互運用データベースシステムの研究開発」の一環として行ったものである．また，OZ＋＋の研究は情報処理振興事業協会（IPA）「開放型基盤ソフトウェア研究開発評価事業」の一環として進めているものである．

参考文献

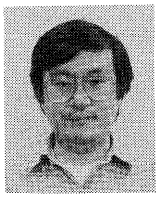
1) Goldberg, A. and Robson, D.: Smalltalk-80, The Language and its Implementation, Addison-Wesley (1983).
2) Liskov, B.: Distributed Programming in Argus, CACM, 31, 3 (Mar. 1988).
3) Chin, R. S. and Chanson, S. T.: Distributed Object-Based Programming System, ACM Computing Serveys, 23 (Mar. 1991).
4) Hoare, C. A. R.: Monitors: An Operating System Structuring Concept, CACM, 17, 10 (Oct. 1974).
5) Andrews, G. R. and McGraw, J. R.: Language Features for Process Interaction, SIGPLAN NOTICES, 12 (Mar. 1977).
6) 塚本他：OZ：オブジェクト指向開放型分散システムアーキテクチャーオブジェクト指向型分散プログラミング言語とその実装，情報処理学会プログラミング言語研究会，89-PL-21-4 (July 1989).
7) 塚本他：OZ：オブジェクト指向開放型分散システムアーキテクチャーオブジェクトの分散管理，情報処理学会シンポジウム「1990年代の分散処理」(Nov. 1990).
8) 塚本，浜崎他：オブジェクト指向開放型分散システムOZ＋の研究開発，電総研彙報，56, 3 (Sep. 1992).
9) 塚本他：オブジェクト指向分散環境OZ＋＋の基本設計，情報処理学会研究報告，93-OS-61-3 (SWoPP 鞆の浦 '93) pp. 17-24 (Aug. 1993).
10) 新部他：OZ＋＋コンパイラによるクラスの版管理，情報処理学会研究報告，94-OS-65 (SWoPP 琉球 '94) pp. 169-176 (July 1994).

（平成7年6月14日受付）



塚本　享治（正会員）

昭和24年生．昭和47年東京大学計数工学科卒業．同年電子技術総合研究所入所，現在分散システム研究室長．知能ロボット用計算機システムの研究の後，オブジェクト指向型分散システム，並列処理システムなどの研究開発に従事．平成元年日本ロボット学会論文賞，平成6年科学技術長官賞研究功績賞受賞．現本会理事．



濱崎　陽一（正会員）

昭和32年生．昭和56年岡山大学大学院工学研究科電気工学専攻修士課程修了．同年電子技術総合研究所入所，以来マルチプロセッサシステム，分散システムなどの研究開発に従事．特に並列・分散アーキテクチャに興味を持つ．現在同所分散システム研究室主任研究官．