システムソフトウェアと
オペレーティング・システム 61－3
（１９９３．８．１８）

オブジェクト指向分散環境 OZ++ の基本設計

塚本 享治[1] 濱崎 陽一[1] 田代 秀一[1] 西岡 利博[2*]
新部 裕[2*] 籠 浩昭[2*] 西山 聡[2] 音川 英之[3*]
吉屋 英二[4*] 鈴木 敬行[5*] 大西 雅夫[6*] 平川 秀忠[7*]

1: 電子技術総合研究所 2: 三菱総合研究所
3: シャープ 4: 富士ゼロックス情報システム
5: シャープビジネスコンピュータ 6: 東洋情報システム
7: 日本ユニシス
*: 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

オブジェクト指向分散環境 OZ++ の基本設計について述べる。OZ++ は、分散アプリケーションの開発 / 稼働を容易にする開発基盤を目指したシステムで、アプリケーションプログラムとオペレーティングシステムの中間に位置する。OZ++ で採用したオブジェクト指向プログラミングを分散環境に対応させた、グローバル / ローカルオブジェクトによるモデル、マルチスレッドによる並行実行のモデルおよび、OZ++ の特徴である管理システムの機能の一部を中心に、基本設計の概要を示すと共に、今後の開発計画と課題についても述べる。

Basic Design of Object-Oriented Distributed Environment OZ++

Michiharu Tukamoto[1] Yoichi Hamazaki[1] Shuichi Tashiro[1] Toshihiro Nishioka[2*]
Yutaka Niibe[2*] Hiroaki Kago[2*] Satoshi Nishiyama[2] Hideyuki Otokawa[3*]
Eiji Yoshiya[4*] Takayuki Suzuki[5*] Masao Oonishi[6*] Hidetada Hirakawa[7*]

1:Electrotechnical Laboratory 2:Mitsubishi Research Institute, Inc.
3:Sharp Corporation 4:Fuji Xerox Information Systems, Co., Ltd.
5:Sharp Business Computer Software, Co., Ltd. 6:Toyo Information Systems, Co., Ltd.
7:Nihon Unisys, Ltd.

*: Researcher of Research, Development and Evaluation of Open Fundamental Software Technology in Information-Technology Promotion Agency, Japan

The basic design of Object-Oriented Distributed Environment OZ++ is described.

OZ++ is a programing environment that makes easy to develop and to execute distributed application programs. OZ++ is positioned between an operating system and application programs.

Two object models of OZ++, the global/local objects model which is an adaptation of object-oriented model to distributed environment and concurrent execution model of objects using multithread execution are shown. Several features of object management systems are outlined.

Future development plan and subjects of research are described also.

1 はじめに

OZ++ は、オブジェクトの交換と共有に基づく分散処理環境である。OZ++ は分散システム全体で統一的に管理されたオブジェクトを情報の単位として用いることにより、アプリケーションを分散環境で容易に稼働させる事ができる分散システムの開発基盤を目指している。

OZ++ は電総研で開発してきた OZ+[1] を基に、実行性能を向上させ、機能の洗練をして、PDS として実用に供する予定である [2]。

現在、OZ++ はその基本設計を終えて、設計の詳細化を進めているところである。ここでは、オブジェクトモデルとオブジェクト管理機構を基本設計を中心に、OZ++ の特徴を述べる。

2 OZ++ システムの概要

OZ++ は、分散アプリケーションの開発 / 稼働を容易にする開発基盤を目指したシステムで、アプリケーションプログラムとオペレーティングシステムの中間にあたる。

2.1 OZ++ の目指すもの

開発を開始するにあたって、OZ++ が目指すシステムは次のようなものである。

1. 分散アプリケーション開発のためのシステムである。
   分散環境において動作するアプリケーションの開発を容易にするように、プログラムのエラーのチェックの強化、支援環境の整備などを計画している。

2. オブジェクト指向に基づいており、ソフトウェア資源の有効利用を援助する。
   既存のクラスおよびクラスライブラリを継承することにより、ユーザが容易にソフトウェア資源を活用する事が出来る。

3. コンパイル時に充分なチェックを行うことにより、実行時のエラーが発生しにくいシステムを目指している。
   言語において型システムをサポートするなど [5]、コンパイル時にチェックが充分に行え、実行時のエラー発生を押える。これは、分散環境においては、実行時エラーは複雑で、再現性が期待できないなど、デバッグを困難にするからである。

4. 不特定多数のユーザが利用することを前提としており、セキュリティについても考慮している。
   セキュリティについては、大規模分散システムには不可欠の要素であり、鋭意設計を進めているところである。

5. システムの機能もアプリケーションレベルの言語で記述されており、拡張性が高い。
   OZ++ の特徴のひとつであるオブジェクトを管理するシステム (次節を参照) 自身が OZ++ の言語で記述 / 実装されているため、拡張が容易で、自由度が高い。

6. 実用的な速度で動作する。OZ++ システムは電総研で開発された OZ+ を基にしているが、OZ+ は速度の点で実用的ではなかった [3]。その経験から、

   - ネイティブコードによる実行
   - 通信機構の簡略化と高速化

   を中心に改良を加え、実用的な速度の実現を目指す。

2.2 OZ++ システムの構成

OZ++ システムの概念的な構成を、図 1に示す。

オブジェクト指向で開発されたアプリケーションは、参照関係をもつオブジェクトの集合から成り、それらの間でメッセージパッシングによりメソッドが起動され、オブジェクトが交換されることにより実行が進んで行く。

OZ++ では、メソッドの実行や通信を行う実行メカニズムの他に、アプリケーションの実行に際して、システム全体で統一の取れた振舞いをサポートするオブジェクト管理機構を設けている。

前者は、個々の計算機に分散され、メソッドの実行や通信などの機械的にその動作が決まるような仕事をし、後者はいろいろな管理のポリシーを含んだものやシステム全体で管理すべき情報の扱いを行う。オブジェクト管理機構はいろいろな種類の管理機構の集まりであり、それぞれはアプリケーションとして実現される。

実行メカニズムとオブジェクト管理機構を分離し、オブジェクト管理機構をアプリケーションレベルで実現することにより、次のような利点が得られる。

1. 実行メカニズムの簡素化
   ポリシーを含まない単純なメカニズムのみを提供するために、機構が簡素に実現でき、それに伴い実行性能の向上が期待できる。

2. システムの拡張性、柔軟性
   アプリケーションレベルでオブジェクト管理機構が実現されているために、変更や拡張が容易である。

管理機構で提供される機能には、次のようなものが考えられる。

- マイグレーション管理
- 複製管理
- 名称管理
- クラス管理
- セキュリティ管理

例えば、あるオブジェクトにメッセージが届いたとき、そのメッセージに従ってメソッドを起動するのは実行メカニズムが行うが、実行すべきメソッドの実行可能インストラクションがその計算機に無かった時に、それを供給するのはオブジェクト管理機構が行う。

オブジェクト管理機構の動作は、アプリケーションからは通常見えないが、アプリケーションの実行においては、陰に陽にオブジェクト管理機構の補助を受けながら実行が進む。

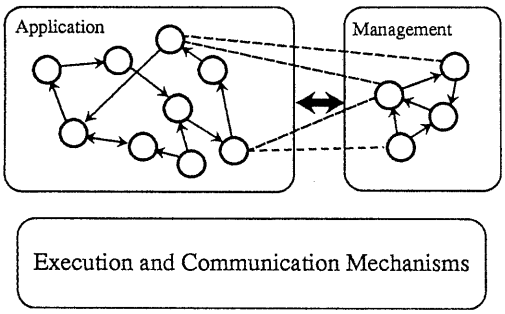

図 1: OZ++ システムの構成

OZ++ システムを実現する実行環境の構成を図2 に示す。

ネットワークで接続されたステーション (計算機) 群の上に実装され、各ステーションには、実行メカニズムであるエグゼキュータ (Executer) が置かれ、オブジェクトはエグゼキュータの中に存在し、エグゼキュータにより実行される。

また、エグゼキュータは他のエグゼキュータとネットワークを介してオブジェクトの交換を行う機構を持っており、これによりオブジェクト間の通信が実現される。オブジェクト交換機構は、オブジェクトのID と通信アドレスの解決機構、オブジェクトの内部表現 (メモリ上の表現) と通信路上の表現とを変換するエンコーダ / デコーダ、通信機構などから成る。

ニュクリアス (Nucleus) は、エグゼキュータの生成 / 消滅 / 管理を行うもので、各ステーションに 1 つ必ず存在する。エグゼキュータは、ステーションに複数個有ってもよいし、一つも無くても良い。

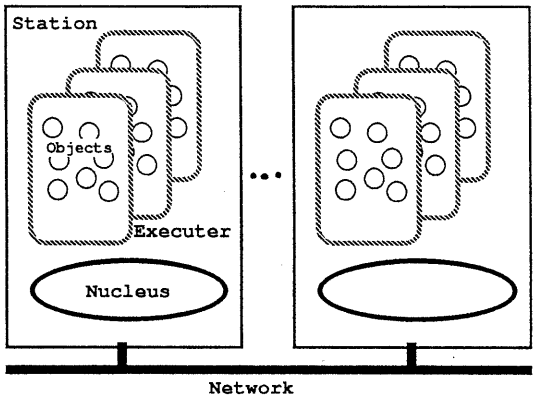

図 2: OZ++ の実行環境の構成

3 OZ++ のオブジェクトモデル

分散を意識しないオブジェクト指向モデルでは、すべてのオブジェクトが等距離でアクセス可能である事が暗黙のうちに仮定されているが、分散環境においては、オブジェクトは複数の計算機に分散しており、そのアクセスも等距離とはならない。

そこで OZ++ では、外部に対してサービスを提供するオブジェクトと、その部品となるオブジェクトの 2 種類のオブジェクトを提供する。前者は他のオブジェクトから分散透過でアクセスされうるのでグローバルオブジェクトと呼び、後者は部品として局所的に使われるのでローカルオブジェクトと呼ぶ。

1 つのグローバルオブジェクトとその部品となる任意個のローカルオブジェクトは、必ず同じエグゼキュータ上に一緒に存在し、これらのオブジェクトの集まりをセルと呼ぶ。

OZ++ におけるオブジェクトモデルを図 3に示す。

3.1 セル

セルはオブジェクトの集合であり, 永続性, マイグレーションの単位である (図中, 破線で囲みで示す)。セルは一つのエグゼキュータ内に複数存在することができる。セル

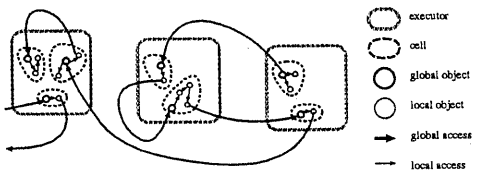

図 3: オブジェクトモデル

は一つのグローバルオブジェクトと任意個のローカルオブジェクトから構成される。

3.2 グローバル / ローカル オブジェクト

OZ++ ではオブジェクトを以下のように二種類に区別して取り扱う。

グローバルオブジェクト 外部から利用され, 共有されるオブジェクト (図中, 太線の円で示す)

ローカルオブジェクト セルの内部で部品として使われるオブジェクト (図中, 細線の円で示す)

ここでオブジェクトのグローバル / ローカルの属性は生成時に決まり, それ以降, 変更することはできない。

グローバルオブジェクトにはシステム全体で一意の ID (以下, これを OID と呼ぶ) が付与され, これによってオブジェクトは識別される。OID を持つということは, システム内の他の全てのオブジェクトからのアクセスに対して開かれているということである。グローバルオブジェクトはセルの窓口である。グローバルオブジェクトはローカルオブジェクトとしての振舞いも兼ね備える。

ローカルオブジェクトはセルの内部で部品として使われるものであり, 外部からは直接参照することができないものである。

オブジェクトは一つのセルに属し, 複数のセルに属することはあり得ない。

3.3 グローバル / ローカル メソッド

OZ++ におけるオブジェクトのアクセスについて図 4に示す。

OZ++ ではオブジェクトのアクセスを以下のように二種類に区別して取り扱う。

グローバルアクセス セル外部からのアクセス (図中, 太線で示す)

ローカルアクセス セル内部からのアクセス (図中, 細線で示す)

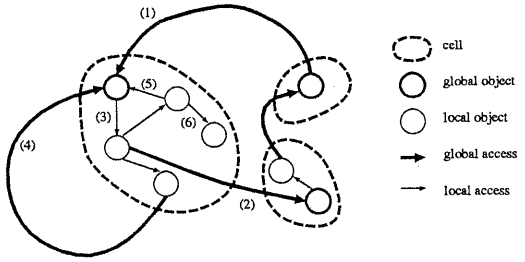

図 4: オブジェクトのアクセス

すなわち, メソッドはグローバルアクセスのメソッドと, ローカルアクセスのメソッドの二種類に分けられる。

オブジェクトのアクセスについて表 1 に示す。表中の括弧内の数字は、図 4の例に対応していることを示す。妥当でないのは、異なるセルではローカルオブジェクトが共有できない事 (A) と、同一セル内にはグローバルオブジェクトが唯一存在するため (B) である。

異なるセル間ではローカルオブジェクトを共有することはできず, ローカルオブジェクトはセル内部からのみアクセスされる。グローバルオブジェクトはセル外部および内部の両方からアクセスされる。

同一セル内であっても外部アクセスとしてグローバルオブジェクトをアクセスすることができる。また, グローバルオブジェクトをローカルオブジェクトとしてアクセスすることも可能である。

表 1: オブジェクトのアクセスの種類

アクセス	source object	destination object	妥当性
異なるセル間	グローバル	グローバル	○ (1)
異なるセル間	グローバル	ローカル	× (A)
異なるセル間	ローカル	グローバル	○ (2)
異なるセル間	ローカル	ローカル	× (A)
同一セル内	グローバル	グローバル	× (B)
同一セル内	グローバル	ローカル	○ (3)
同一セル内	ローカル	グローバル	○ (4),(5)
同一セル内	ローカル	ローカル	○ (6)

3.4 プロセス

OZ++ においてオブジェクトは計算機構を提供する資源およびその状態であり, その手続きの実行 (計算) とは分離している。OZ++ ではスレッドのメソッド呼び出しの連鎖をプロセスと呼び, 計算の抽象としてこれを扱う。

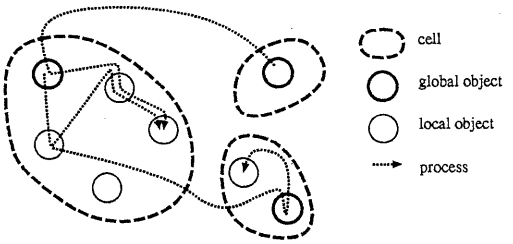

図 5: プロセスとオブジェクト

このプロセスとオブジェクトの関係を図5に示す。図中, プロセスは点線で示す。

OZ++ では, 複数のスレッドが同じオブジェクトのメソッドを同時に実行できる。プロセスはオブジェクトの中を手続き呼び出しの意味で自由に実行できる。同期 / 排他制御の機構として, モニタを用意している。オブジェクトはモニタとすることができ, 異なるスレッドは条件変数で同期をとることができる (次節参照)。

メソッド起動時にフォークすることを指定すると, 新たなプロセスが生成され、元のプロセスと並行にマルチスレッドによる実行が進む。つまり, プロセスの生成は呼出側の制御による。

プロセスに対しては, 終了を待ち結果を得る join, 分離する detach, 実行を中断する abort などの操作が用意されている。

3.5 クラスシステム

すべてのオブジェクトは何らかのクラスのインスタンスであり, どのクラスのインスタンスであるかの情報を持つ。そして, その振舞いはクラスによって規定される。

OZ++ は多重継承のクラスシステムを持つ。継承の機構により, 親クラスはその子孫に共有される。多重継承により生じる名称の衝突は、改名により解決する。

クラスは, 親クラス, メソッドのシグネチャおよびインストラクション, インスタンス変数の型およびその外部表現のための情報 (リプレゼンテーション情報) を持つ。

オブジェクトのグローバル / ローカルの性質はクラスによって規定されるのではなく、オブジェクトの生成時に決まる。その意味で、クラスはグローバル / ローカル両方のオブジェクトを生成する雛型である。

また、OZ++ 言語ではクラスは静的な概念であり, クラスをオブジェクトとして扱うことはできない。

4 OZ++ の実行モデル

4.1 スレッドによる並列実行と排他制御 / 同期

OZ++ では 1 つのオブジェクトのメソッドを実行する複数のスレッドが同時に走り得る。これによりメソッドは並行実行を行なうことができる。スレッドのメソッド起動の連鎖がプロセスである。スレッドはグローバルメソッドの起動時に生成される。ローカルメソッドの起動は手続き呼びだしで実行される。

スレッド間の排他制御を行なうオブジェクトはモニタであり, スレッド間の同期は条件変数によって行なう。

モニタのオブジェクトでは各メソッドは, ロックを得るか否かの属性を持つ。一つのオブジェクトに対して, ロックを必要とする最大 1 個のメソッドとロックを必要としない任意個のメソッドが同時に実行され得る。

4.2 オブジェクトのコピーイン / コピーアウト

ローカルオブジェクトは, セル間で共有されることはなく, セル間でローカルオブジェクトへの参照が受け渡されることはない。セル間を渡る時には, ローカルオブジェクトはそれが参照するローカルオブジェクトと共にコピーされる。

グローバルメソッドの起動時, 引数のローカルオブジェクトは呼出側でコピーアウトされ, 呼び出された側にコピーインされる。グローバルメソッドの終了時も同じく返り値のローカルオブジェクトがコピーイン / コピーアウトされる。

グローバルオブジェクトは、その参照である OID が受け渡される。

4.3 メソッドの動的な束縛 (多相性)

OZ++ 言語において変数は静的な型 (クラス) を持つが, メソッドの実行は動的な型 (静的なクラスの子孫である) によって行なわれる。これによって変数は一つ以上のクラスのオブジェクトを参照する多相性をもつ。

ただし, メソッドのグローバル / ローカルの違いはシグネチャで区別され, 継承した側で変更することはできないので, 動的にメソッドのグローバル / ローカルの属性が決まるということはない (コンパイル時に決定される)。

5 オブジェクト管理機構

OZ++ では、オブジェクトの基本的な振舞いの管理を行なう部分についても、積極的にオブジェクトの機能として実現し、エグゼキュータを軽くする方針が採用されている。本節では、このようなオブジェクト管理機構のうち、オブジェクトの基本的な動作に密接に関わる部分について、その概要を示す。

5.1 オブジェクトマイグレーション

すべてのプロセスが同一計算機上で動作する環境と異なり、分散環境においては、長時間停止することの許されないサービスをどのように計算機のトラブルなどから守るかが問題となる。この問題を解決する手段の一つとして、OZ++ ではオブジェクトマイグレーションの機能を提供している [4]。

オブジェクトマイグレーションを実現する際の問題は以下の二点である。

- マイグレートしてしまったオブジェクトに対する通信はどのように行なうか。
- 実行中のスレッドや、排他制御用の情報 (モニタロックや条件変数など) はどうなるか。

前者については、[4] で述べたので、ここでは後者について述べる。

一般に、スレッドの動的な状態を保存しながら別の計算機に移動することは (特にアーキテクチャが異なる場合) 難しい。しかし、マイグレーションすることで実行中の処理が中途半端な状態で中断されてしまうのではマイグレーションの目的を果たすことができない。OZ++ では、オブジェクトの停止と再起動の仕組を提供する。プログラマは、この機構を用いてスレッドの状態を保ったマイグレーションをプログラムすることができる。

マイグレーションは次の手順で行なう。

1. オブジェクトを停止する。
2. モニタロックや条件変数を、送受信の際のエンコード / デコードでクリアする。
3. オブジェクトのデータ部分を移動する。
4. オブジェクトを再起動する。

ここで、オブジェクトの停止と再起動は、次のようなメカニズムを利用する。

- オブジェクトの停止
  オブジェクトを停止させるときは、そのオブジェクトの **freeze** メソッドを呼び出す。各クラスは freeze メソッドを実装し、その中で実行中のスレッドを停止しなければならない。
- オブジェクトの再起動
  オブジェクトを再起動させるときは、そのオブジェクトの **wakeup** メソッドを呼び出す。各クラスは wakeup メソッドを実装し、その中で必要な回復処置を取らなければならない。

デフォールトの freeze メソッドは全てのスレッドの終了を待つメソッドであり、デフォールトの wakeup メソッドは、何もしないメソッドである。

5.2 永続オブジェクト

オブジェクトのデータを二次記憶に維持する機能をオブジェクト自身の責任で実装させるのは、オブジェクト ID を維持しにくいという点で問題がある。OZ++ では、オブジェクトを不変のオブジェクト ID で二次記憶に維持する機能が提供される。このように維持されるオブジェクトを永続オブジェクトと言う。

OZ++ では、エグゼキュータごとに一つずつ、オブジェクトマネージャと言うオブジェクトがある。永続オブジェクト管理はオブジェクトマネージャの機能である (一部エグゼキュータの機能を利用する)。

永続オブジェクトを管理する場合に問題となるのは以下の二点である。

- 実行中のスレッドや、排他制御用の情報はどうなるか。
- 二次記憶上のオブジェクトに対する通信はどうなるか。

前者については、前項の方針を踏襲する。すなわち、オブジェクトの停止と再起動はプログラマの責任とし、排他制御用の情報はエンコーダ / デコーダがクリアする。

後者については、二次記憶上のオブジェクトに対してメッセージ送信が行なわれると、オブジェクトマネージャはそのオブジェクトを直ちに主記憶に展開し、wakeup メソッドを呼び出してからそのメッセージを処理させる。

5.3 複製管理

分散環境においては可用性は重要な要素である。複製管理はオブジェクトの可用性を向上させる最も重要な技術で

あり、多くの方式が提案されている[6]。しかし、普遍的に最適な方式はなく、目的に応じた選択が必要である。

OZ++では、言語機能としては複製を提供しない。アプリケーションの対象とする分野により、時間制限の厳しいものや、高信頼性を要求するものなどがあるため、言語機能とするには選択の幅があまりにも広いためである。その代わり、主要な複製管理方式をいくつか選択的に使用できるようにクラスライブラリを用意する。プログラマは利用したい複製管理方式を提供するクラスを継承または利用し、その複製管理方式で要求されるコンベンションを満たすように作られたテンプレートにしたがってコーディングすることによって、典型的な複製管理のいくつかが、複雑な一貫性維持プロトコルを設計することなく実現できる。

複製機能はOZ++システム自身の実装においても重要な役割を果たす。このため、認証やアカウントの管理などに利用されるような、高度の可用性を提供する複製管理機能もライブラリとしてサポートされる。一般のプログラマも必要ならばその機能を用いることができる。

複製されたオブジェクトに対する通信については、[4]に述べられている。

5.4 分散トランザクション管理

分散トランザクション管理は、離れた場所にあるデータに対する並行的な操作が、安全に排他的に行なえるようにするための仕組みであり、分散環境の構築にはなくてはならない要素技術である。OZ++では、複製管理と同様の理由で、言語機能としてはトランザクションを提供しない。そして、いくつかの主要なトランザクション管理方式を選択的に使用できるようなクラスライブラリが用意される。プログラマは利用したいトランザクション管理方式を提供するクラスを継承または利用し、そのトランザクション管理方式で要求されるコンベンションを満たすように作られたテンプレートにしたがってコーディングすることによって、典型的なトランザクション管理のいくつかを、複雑なロッキングプロトコルやデッドロック検出の設計をすることなく実現できる。

5.5 クラス名管理

小規模の環境と異なり、(特にマルチユーザの)分散環境では、クラス名が容易に衝突してしまうため、これを大域的にユニークなものとして扱うことは難しい。OZ++では、この問題を解決するため、クラス名と実際のクラスとの対応表(スクールと言う)を環境中に維持する。この役にあたるオブジェクトを**スクールサーバ**という。OZ++では、クラス名の指定に際してスクールを指定することができ、出所の異なるライブラリを同時に利用してもクラス名の衝突が起きないようにすることができる。

5.6 クラスの版管理

分散開発環境では、クラスの異なる版が同時に利用可能でなければならないことがしばしばある。OZ++ではクラスバージョンサーバというオブジェクトによって異なる版のクラスが管理される。

前項で述べたスクールは、クラス名に対応してクラスの特定の版を指定するのにも用いることができる。この機能を用いて、開発中のソフトウェアを構成するクラスのうち、安定している部分を選択的に利用することが可能である。

クラス名に対応するクラスがすべて特定の版に固定されているようなスクールをモジュールと言う。ソフトウェアはモジュールを公開することでリリースできる。スクールにはモジュールを取り込む機能があり、利用者は、自身のスクールに、利用するソフトウェアのモジュールを取り込むことによって、リリース版のソフトウェアを利用することができる。

分散開発環境では、特に再コンパイルすることなしに最も新しい公開版が利用できることが望ましい。このため、クラスの実行可能コードは通常、利用するクラスのバージョンを動的に決定するようになっている。これに対してモジュールに含まれる版は、他のクラスの特定の版を利用することが分かっているので、静的な版決めによる最適化が可能である。

6 まとめ

現在開発を進めているオブジェクト指向分散環境OZ++の基本設計について、そのオブジェクトモデルとオブジェクト管理機構を中心に述べた。

OZ++は、グローバル/ローカルオブジェクトのモデル、複数のプロセスによる並行実行により、オブジェクト指向プログラミングを分散環境へ適応させ、オブジェクト管理機構によってシステム全体で統一的なオブジェクトの管理を可能にしている。

OZ++では、オブジェクト間で受け渡されるメッセージもオブジェクトであり、その意味で純粋なオブジェクト指向分散システムである。これは、OMGのCORBAなどのようにオブジェクト間で通信されるものがオブジェクトとは限らないシステムとの本質的な相違である。

このプロジェクトにおいて、開発を予定しているのは、次のようなものである。それぞれの概要と開発計画を簡単に記す。

分散オブジェクト指向言語 (OZ++ 言語) 言語の基本設計については [5] で述べた。現在、言語とコンパイラ、リンカなどの言語処理系の詳細設計を進めており、オブジェクト管理機構が開発されるまでの間、オブジェクト管理機構の機能をエミュレートした第 0 版の言語処理系は、今年度末の完成を予定している。

実行メカニズム マルチスレッドのスケジューラ、メモリ管理、I/O 管理などのオブジェクトの実行に関する部分と、オブジェクトの交換など通信に関する部分の 2 つに分けて、エグゼキュータおよびニュクリアスの詳細設計とコーディングを進めている。オブジェクト管理機構が開発されるまでの間、オブジェクト管理機構の機能を既存の分散ファイルシステムなどでエミュレートした第 0 版の実行メカニズムは今年度末の完成を予定しており、言語処理系と共に、OZ++ 言語で記述したプログラムが実行可能となる予定である。

オブジェクト管理機構 オブジェクト管理機構のうち、オブジェクトの基本的な動作に密接に関わる部分から順次設計を進めており、OZ++ 言語によるコーディングに着手するところである。今年度末には、コーディングされた管理機構のプログラムを実行出来る環境が整う予定であるので、順次動作させ、それに合わせて言語の処理系、実行メカニズムを更新してゆく。

開発支援システム ユーザの OZ++ 言語による分散アプリケーションの開発を支援するもので、クラスブラウザ、デバッガなどのツール群である。OZ++ 言語による開発が本格化するのに合わせて、開発を進める予定である。

また、基本設計では検討が不十分であった次のような点については、今後の課題である。

- スタートアップ手順
  OZ++ は実行メカニズムとオブジェクト管理機構の両者の動作により、アプリケーションの実行が行われる。しかしながら、システムの立ち上げ時などには、システムの動作していない状態から、各部を徐々に起動して、完全に立ち上がった状態へ移行する必要が有り、途中の不完全な状態においても安定に動作する仕組が必要となる。

- 大規模ネットワークへの対応
  複数のネットワークセグメントから成る大規模ネットワークや、広域網の介在する環境への対応。ネットワークセグメント間の通信や、その管理方式などについての検討が必要である。

- セキュリティ管理
  開放型のシステムにおいては、不特定多数のユーザの利用が予想されるために、セキュリティの重要性は高いが、同時に過大なオーバヘッドがあってはならない。現在

- 異機種計算機環境開発には、Sun microsystems の Sparc Station を LAN 接続したものを使用しているが、その他の機種、例えば IBM-PC with Windows NT への移植、異機種計算機環境への対応についても、検討している。

  オブジェクトの交換については、OZ+ において実証済みであるが、実行可能インストラクションの機種別供給などの仕組が必要である。

本研究で開発されたプログラムなどの成果物は、研究開発プロジェクトの終了する平成 8 年 3 月をめどに PDS として公開する予定である。

熱心な討論を頂いた、藤野晃延 (富士ゼロックス情報システム)、千葉滋 (東京大学理学部) 両氏に感謝する。

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] 塚本他：「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、vol. 56、No. 9、Sep. 1992

[2] 塚本他:「OZ++ 開発計画」、情報処理学会第 45 回全国大会、Oct. 1992

[3] 濱崎他：「OZ+ のオブジェクト通信機構の評価」、情報処理学会第 45 回全国大会、Oct. 1992

[4] 濱崎他:「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

[5] 西岡他:「オブジェクト指向分散環境 OZ++ の言語の基本設計」、情報処理学会第 46 回全国大会、Mar. 1993

[6] San, H.S.: "Replicated Data Management in distrubted Database Systems," it Sigmod Record, Vol. 17, No. 4, pp 62–69, Dec. 1988.