オブジェクト指向分散処理環境 OZ++の実行機構の設計

6F－4

濱崎 陽一 (電子技術総合研究所)、吉屋 英二 (富士ゼロックス情報システム *)、
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 *)、
平川 秀忠 (日本ユニシス *)、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処理環境OZ++は、電総研で開発されたOZ+[1] を元に、分散アプリケーションプログラムの開発を容易にする環境を目指して開発を進めており、基本設計を終えて設計の詳細化および開発を進めているところである [2]。

ここでは、OZ++のオブジェクトの実行 / 交換メカニズムを提供する実行機構であるエグゼキュータについて述べる。

2　OZ++の構成

OZ++はシステム全体で管理されたオブジェクトによる分散処理環境を実現するものであり、オブジェクトの管理もオブジェクトによって実現される [3]。

アプリケーションであるオブジェクトおよび管理オブジェクトの実行メカニズムがランタイムカーネルである。ランタイムカーネルはオブジェクトの実行とメッセージ交換のメカニズムであるエグゼキュータと、エグゼキュータの生成 / 管理を行うニュクリアスから成る。

エグゼキュータは、オブジェクトの生成 / 管理を行い、またそのメソッドの実行を行う。そのため、エグゼキュータは各種のランタイムルーチンを持ち、管理オブジェクトからロードした実行に必要な情報であるクラス情報、レイアウト情報ならびに実行可能コードを提供する。

クラスの実行可能コードのようにシステムのオブジェクトで管理される情報は、エグゼキュータとそうしたオブジェクトが直接やり取りするのではなく、エグゼキュータに必ず一つ存在するオブジェクトマネージャ (以下 OM と略す)[4] を介して管理システムから得る。

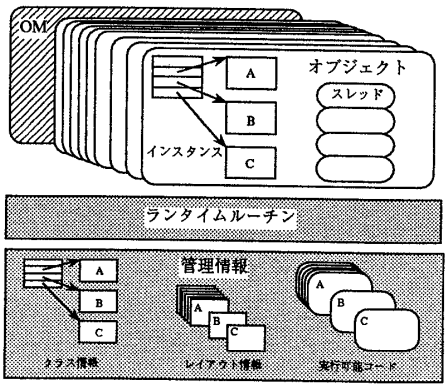

図 1: エグゼキュータ上のオブジェクトの概念図

3　管理情報

実行に際して、必要な情報がそのエグゼキュータ上に存在しなければ、必要に応じてロードする。情報が揃うまで、実行のスレッドは止められる。実行に必要な情報には、次のようなものが有る。

実行可能コード 実行可能コードは、メソッドの実行に必要なメソッドのコードで、OZ++言語をコンパイルして得られる。エグゼキュータにロードする際に、エグゼキュータの持つランタイムルーチンとダイナミックリンクされる。メソッドのロードは、クラス毎に行われる。

クラス情報 OZ++のクラスは多重継承であり、各クラスはバージョンシステムを持っている。そうしたクラスに関する情報はクラスシステムを管理するオブジェクトであるスクール、クラスバージョンズ、クラスから提供される。

そのクラスのバージョンおよび継承するクラスのバージョンは、オブジェクトのインスタンシエート時に決定される。そのクラスおよび継承する全

A Design of Execution Mechanisms of OZ++ : An Object-Oriented Distributed Processing Environment
Yoichi Hamazaki(Electrotechnical Laboratory),
Eiji Yoshiya(Fuji Xerox Information Systems, Co., Ltd.*),
Masao Onishi(Toyo Information Systems, Co., Ltd.*),
Takayuki Suzuki(Sharp Business Computer Software, Co., Ltd.*),
Hidetada Hirakawa(Nihon Unisys, Ltd.*),
and Michiharu Tsukamoto(Electrotechnical Laboratory)
*: Researcher of Evaluation of Open Fundamental Software Technology Project in Information-technology Promotion Agency, Japan

てのクラスのバージョンの組をコンフィギュレーションと呼ぶ。

クラス情報は、コンフィギュレーションと対応したオブジェクトの構成の情報であり、自身と継承するクラスのバージョンと、メソッドの実装クラスの情報を持っている。クラス情報は、オブジェクトの生成やメソッドを起動する際に使われる。

レイアウト情報 レイアウト情報は、オブジェクトがメモリ上でどのように表現されているかを示すもので、オブジェクトをエグゼキュータ間で交換したり、二次記憶上に永続化するために必要となる。

レイアウト情報は、計算機のアーキテクチャ毎に提供され、異機種間でのオブジェクトの交換の際には、双方のレイアウト情報を参照する。

4 オブジェクトのインスタンス

OZ++のオブジェクトには、システムでユニークな名称を持ち外部からアクセスされうるグローバルオブジェクトと、その部品であるローカルオブジェクトの2種類がある。

各グローバルオブジェクトは専用のヒープを持っており、その上にグローバルオブジェクトおよびそのグローバルオブジェクトが持つローカルオブジェクトの実体が割り当てられる。グローバルオブジェクトへの参照はその名称であるOIDにより行われる。グローバルオブジェクト間でローカルオブジェクトの共有は生じないので[2]、ポインタによる相互参照はヒープの中で閉じており、グローバルオブジェクトを単位としてGCを行うことが出来る。

OZ++は多重継承をサポートしており、インスタンスはそのオブジェクトのクラスおよび継承するクラスの部分の集合体となる。

グローバルオブジェクトの生成はOMに要求することにより行われ、そののち、初期化のための特殊なメソッドであるコンストラクタの実行により初期化される。

5 オブジェクトの実行

メソッドの実行はスレッドにより行われ、一つのオブジェクトで複数のスレッドの実行が可能である。スレッド間の同期はモニタにより行われる。

グローバルオブジェクトのメソッド呼び出しでは、呼び出されたグローバルオブジェクトにメソッドを実行するスレッドが生成され、呼び出した側のスレッドは結果が返されるまでブロックされる。呼び出されるオブジェクトが別のエグゼキュータの場合には、通信機構によりデータの授受が行われる。こうしたグローバルオブジェクトのメソッド呼び出しは、チャネルと呼ぶADTで抽象化されており、オブジェクトの位置透過性を実現している。

また、ローカルオブジェクトのメソッドは、それを呼び出したスレッドの延長で実行される。

6 OMとのインタフェース

OMの機能の一部は、ランタイムルーチンの呼び出しにより実現されており、そのためのランタイムルーチンが用意されている。

また、エグゼキュータからOMを介したサービスを要求するアップコールは、各要求ごとに存在するOMのデーモンプロセス(スレッド)によって処理される。これは、アップコールの度に実行コンテクストを生成するオーバヘッドを避けるためである。

7 まとめ

OZ++の実行機構である、エグゼキュータについて述べた。エグゼキュータは実行のメカニズムであり、ポリシーを含むような上位の管理は特殊なオブジェクトであるOMとシステムに散在する管理オブジェクトにより行われる。これにより、拡張性が高く、変更が容易な構成に成っている。

現在、詳細設計を終り、実装を進めているところである。

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: 「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)、Aug. 1993

[3] 籠他: 「オブジェクト指向分散環境OZ++のオブジェクト管理機構の概要」、情報処理学会第47回全国大会、Oct. 1993

[4] 大西他: 「オブジェクト指向分散処理環境のオブジェクトマネージャの設計」、情報処理学会第48回全国大会、Mar. 1994

[5] 吉屋他: 「オブジェクト指向分散環境OZ++のクラス管理方式」、情報処理学会第47回全国大会、Oct. 1993