オブジェクト指向OS Apertosを用いた分散環境の実現*

6C−1

山下 貴典† 横手 靖彦‡ 岡村 英明§ 砂原 秀樹† 尾家 祐二†
† 奈良先端科学技術大学院大学 ‡ ソニー(株)
§ (株)ソニーコンピュータサイエンス研究所

1 はじめに

これからのコンピュータネットワークの形態として、携帯型端末、ゲーム機、NC、VOD System等の計算資源の限られた組み込み型機が接続される事が予想される。このようなネットワーク上では

- 頻繁なシステム構成の変更
- 制限された計算資源での快適な実行
- 位置透過性、移動透過性の実現

などに対応できる新しいソフトウェアアーキテクチャが必要となる。

SONY CSLで開発中のOS Apertosは、オブジェクト指向基づいたアプローチによりこのような環境を実現しようとしている[1]が、Apertosによる分散環境への移行を促すには既存のユーザ環境の共存と開発環境の充実が欠かせない。

本稿では分散環境実現のための手法としてのオブジェクト指向とそれに対するApertosのアプローチについて紹介し、上述の課題を解決する手段の一つとしてのApertos Emulatorとその実装法について述べる。

2 オブジェクト指向による分散環境

ネットワークを複数の抽象化レベルで階層化しネットワーク全体を一つの計算場としてユーザに見せる手法として、オブジェクト指向を適用することが考えられている[5]。

ここでいうオブジェクト[1]とは個々の計算を行う活動的な主体であり，オブジェクト指向とはそのようなオブジェクト同士のメッセージの交換により計算を表現する事である。さらにそのオブジェクトを複数の抽象化レベルにより階層化し、オブジェクト同士が互いの結び付きを動的に変られるようにする事で（図1）オブジェクトの振舞そのものを状況に応じて変化させる事が可能になる[3]。

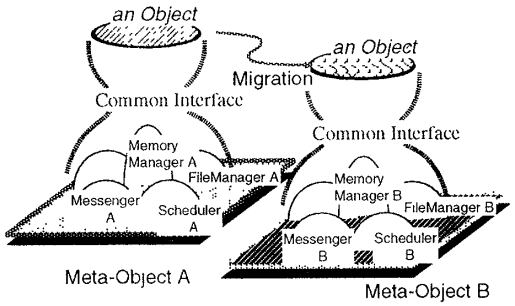

図1: オブジェクトの自己反映モデル

このような自己反映可能な特性は、システムの構成、振舞を起動時に予め予測することが困難な大規模な分散環境への適用に向いている。そして、その自己反映可能な機構を利用してシステムが備える様々なレベルでの抽象化を定義/利用できるようにしたのがApertosである。

3 Apertos Emulator 実装

以上述べたようにApertosは分散環境の実現手段として有効なOSであるが、実際にネットワーク上の個々の計算機にApertosのオブジェクトの実行環境が存在する事が大前提となる。

そこでそのようなApertosオブジェクト実行環境を既存のシステム上に提供しApertosによる分散環境を実現するための第一段階として、UNIX上でのApertos Emulatorの実装を行う。

Apertos EmulatorはApertos Applicationの開発環境の充実と既存の計算機環境との親和性を図ること目標とし、以下のような機能を提供する。

- Apertos objectの実行 (binary互換)
- debug用インタフェースの提供
- Apertos Emulator上でのアプリケーション開発
- Apertso - Apertos Emulator間でのmessage passingとobject migration[1]の実現

Apertos Emulatorの実装における最初の課題はUNIX上でいかに並行オブジェクトの実行環境を作るかである。

現在のApertosでのオブジェクトは、図2のようにオブジェクトの一つ一つをプロセスの仮想空間に割り

*Distributed Computing Enviorment with The Apertos Reflective Operating System
†Nara Institute of Science and Technology
‡SONY Corporation
§SONY Computer Science Laboratory Inc.
[1]オブジェクトはそれが表す計算の抽象度の違いによって“base”と“meta”に分離される[2]

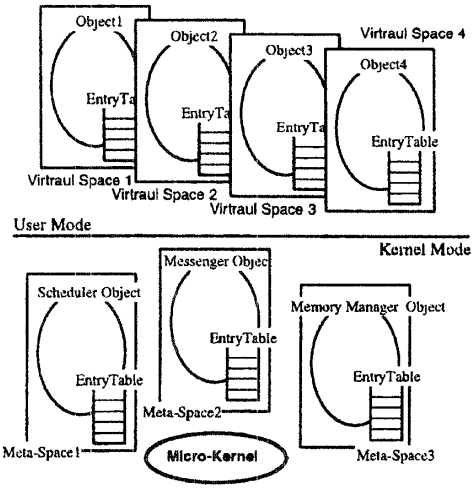

図 2: Apertos 上のオブジェクト

当てている。[2]オブジェクトの各メソッドはエントリーテーブルに登録され、オブジェクトへの処理の依頼はこのメソッドのエントリーに対して行われる。

これを UNIX 上で実現させるための手段として本実装ではユーザレベルでのマルチスレッド機構を用いる。

Apertos Emulator では、各オブジェクトの実行をプロセス内のスレッドに一つ一つ割り当てることで UNIX 上に並行オブジェクトを実現する。図2でも示したように各オブジェクトの処理はメソッドのエントリーテーブルを通して依頼される。そこで Apertos Emulator では、multi-thread[4] で thread として実行する関数をオブジェクトのメソッドのエントリーとして実行できるようにしている。これにより、UNIX 上で並行オブジェクトの実行環境が実現される。

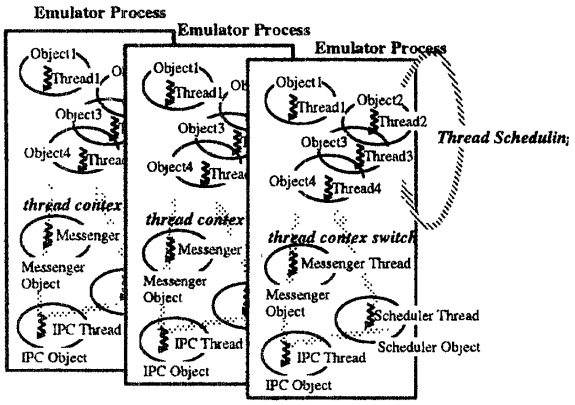

図 3: multi-thread による並行オブジェクトの実装

アプリケーションオブジェクトはスケジューラオブジェクトによりスケジューリングされる。また、アプリケーションオブジェクトが他のオブジェクトとメッセージを交わす時は、他のオブジェクトの実行を表すスレッドはブロックされる。代わりに Messenger Object や IPC Object のメソッドを表すスレッドが起動されオブジェクト間の通信を行う。これは、オブジェクト間の通信/スケジューリングといったより抽象度の低い計算が行われる部分を meta-object というオブジェクトとして分離する Apertos のオブジェクトモデルに従っているためである。

またオブジェクトの自己反映性の実現のしくみであるオブジェクトの移送も Apertos Emulator 間においてで可能である。これは アプリケーションオブジェクトのバイナリイメージと そのオブジェクトの実行を表している thread のスタックとコンテキストを stream として流す事で実現できる。

これにより、例えばオブジェクトのスケジューリングポリシーを変更したい時は、別のスケジューリングが行われている Emualtor プロセスへそのオブジェクトを移送させることでスケジューリングの変更をオブジェクトの実行を止める事無しに実現できる。

ユーザ側からはこれらの操作は隠されており、ユーザは、通信相手のオブジェクトが同じプロセス内に存在するのか、別のホストの Emulator プロセスに存在するのか、あるいはアプリケーションオブジェクトがディスクからロードされているのか、ネットワーク経由でロードされているのか等を意識することはない。

4 まとめ

本稿では Apertos による分散環境の実現性と Apertos Emulator の必要性、その実装法について述べた。

現在の実装はまだ Apertos としてのごく基本的な機能を実現できたのみで、今後さらなる system object の拡充、debug 機能の追加、他のプラットホームへの移植を図り、Apertos 環境を広げて行く必要がある。

参考文献

[1] Yasuhiko Yokote: "The Apertos Reflective Operating System: The Concept and Its Implementation", SCSL-TR-92-014(1992).

[2] Yasuhiko Yokote: "Kernel Structuring for Object-Oriented Operating System: The Apertos Approach", SCSL-TR-93-014,(1993).

[3] Takao Tenma, Yasuhiko Yokote, Mario Tokoro: "Implementing Persistent Objects in the Apertos Operating System", SCSL-TR-92-014,(1992).

[4] David Keppel: "Tools and Techniques ofr Building Fast Protable Threads Packages" Technical Report UWCSE 93-05-06, (1993).

[5] 所, 松岡, 垂水: "オブジェクト指向コンピューティング", 岩波書店, (1993).

[2]全てのシステムオブジェクトが kernel mode で動作するとは限らない。