3K−4

分散オペレーティング・システムISEとその並列処理言語について

宮川 祐史　　太田 義勝　　大山口 通夫

(三重大学 工学部)

1 はじめに

近年、コンピュータ・システムはワークステーションやスーパーコンピュータなどの計算機を多数、高速なネットワークで結合して利用する分散システムに移行しつつあり、このような分散環境に適したオペレーティング・システム(分散OS)の開発が求められている。分散OSとは、ネットワークで繋がれた計算機をネットワーク透過なものとし、まるで1台の計算機のように見せることにより、CPUをはじめとするさまざまな資源を有効利用させるものである。これまで、分散OSとしてV[1],ToM[2],Clouds[3],Amoeba[4]などが提案されている。

我々の研究室では、オブジェクト指向を取り入れた分散オペレーティング・システムISEを研究・開発している。ISEの特徴としては、

1. 分散システムに適した計算モデルであるオブジェクト／スレッド・モデル
2. 計算モデルの記述に適した並列処理言語 Parallel Objective-C
3. コア／システム／ユーザ領域の3層構造からなるシステム構成

があげられる。本稿では、これらの特徴について報告する。

2 オブジェクト／スレッド計算モデル

ISEでは、異なったアーキテクチャを持つ計算機からなる分散システムを対象にしており、その計算モデルとしてオブジェクト／スレッド・モデルを採用した。この計算モデルは、分散システムのリソースをオブジェクトとして、プロセッサをスレッドとして、抽象化したものである。オブジェクト間のメッセージ・センディングは、制御の流れであるスレッドがオブジェクト間を渡り歩くことで実現される。

ISEでは、プログラムに相当するものをProgram Objectとして抽象化し、その中には複数のオブジェクトと複数のスレッドが存在する。

分散システムでは多数のプログラムが協力しあって仕事を進めることが重要である。本研究では、複数のProgram Objectの集まりをGroup Objectとして抽象化した。同じGroup Objectに属するProgram Object間では、他のProgram ObjectのFactory Object (クラス) のInstance Objectを生成することを許している。この機能により、各Program Objectは他のProgram Objectの資源を有効利用することができる (図1)。

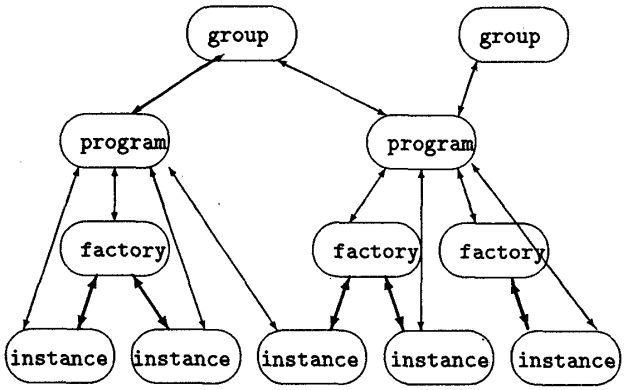

図1. Program Object と Group Object

3 並列処理言語 Parallel Objective-C

分散システムにおいては、計算モデルを明確に表現でき、かつ、並列処理を容易に記述できる並列処理言語の設計が重要である。そこで、我々は、オブジェクト／スレッド・モデルを明確に表現できるプログラミング言語として、Parallel Objective-Cを提案する。これは、従来のObjective-C[5]を基本にして以下の機能を追加したものである。

(1) 拡張メッセージ呼び出し

[*factry_object@program_object message*]

他のProgram Objectに属しているFactory ObjectのInstance Objectを生成するためのメッセージ呼び出し。

(2) 並列実行文

par{$statement_1$ $statement_2$ $\cdots$ $statement_n$}

On the Distributed Operating System ISE and its Parallel Processing Language.
Yuji Miyagawa, Yoshikatsu Ohta and Michio Oyamaguchi
Faculty of Engineering, Mie University

各文は並列に実行し、全ての文が終了した時、並列実行文が終了する。

(3) 並列実行式

id t; t = [*obj selector arg*] $;

...

s = [t recieve];

式(メッセージ呼び出し)の後に$を付けることにより、その式の処理が終了を待たずに"Thread"クラスのオブジェクトの値を返す。その"Thread"クラスのオブジェクトにreceiveメッセージを送ることで待ち合わせを行い、式の結果を得る。

(4) ロック機能

あるオブジェクトに複数のメッセージ呼び出しが同時に行われる時には、1つのメッセージ呼び出しが選ばれて実行され、他のものは待たされる。但し、宣言により同時に実行させることも可能である。

(5) クラス変数

Instance Object 間でデータを共有するためにクラス変数を導入した。

4 ISEのシステム構成

ISEはハードウェアに近い順にコア領域、システム領域、ユーザ領域の3層から構成されている。コア領域は、各マシン毎に存在するCOREからなり、各COREは、そのマシンでのスケジューリングやデバイスの管理など従来のOSのカーネルがしていた仕事を行う。システム領域は、ネットワーク透過なサービスをユーザに提供するものであり、ファイル・サーバやウィンドウ・サーバなどのシステムのProgram Objectから構成されている(図2)。

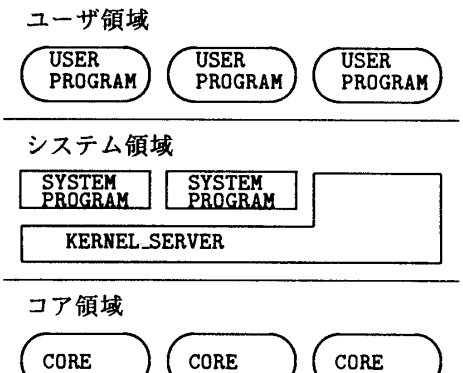

図2. ISEのシステム構成

カーネル・サーバはシステム領域に存在し、ネットワーク・ワイドにオブジェクト、デバイス、スレッドなどの管理を行う。これは、ユーザ・プログラムがホストネームを一切気にすることのない、完全なネットワーク透過性を実現するために必要である。コア領域との直接のやりとりは、カーネル・サーバだけが行う。ユーザ領域は、ユーザのProgram Objectからなる。ユーザ領域とシステム領域はSYSTEMというGroup Objectを通して、オブジェクト指向の枠組で、計算機システムの資源を利用できるように設計されている。カーネル・サーバからユーザ領域までは、すべてParallel Objective-Cで記述される。

ISEでは、シェルのようなユーザ領域のプログラムから、ユーザのProgram Objectを新たに生成するには、カーネル・サーバ中のPROGRAM_OBJECTという名前のFactory Objectへcreateメッセージを送ることで実現される。これは、Parallel Objective-Cで次のように記述される。

pob=[PROGRAM_OBJECT@KERNEL_SERVER create:"*USER_PROGRAM* "];

このシステムコールが行なわれた時、カーネル・サーバは、そのユーザのProgram Objectに含まれている各Factory Objectを適当なマシンへ分散配置し、そのことによって、分散環境の資源が有効利用される。この式の返り値に、runメッセージを送ること(すなわち、[pob run];)で、そのProgram Objectのmain関数からスレッドが動きはじめる。

なお、Parallel Objective-Cのソースをコンパイルすると、Factory Object毎に、ネットワークにつながっている機種の種類だけ、実行型ファイルが生成される。

5 おわりに

本稿では、異なったアーキテクチャを持つ計算機からなる分散システムのオペレーティング・システムとしてISEを提案し、その計算モデルとしてオブジェクト/スレッド・モデル、並列処理言語としてParallel Objectve-Cについて述べた。さらに、コア/システム/ユーザ領域の3層構造からなるシステム構成について述べた。

謝辞 日頃、御指導頂く三重大学 那須正和教授、鎌部浩助手、並びに熱心に討論して頂いた研究室の諸氏に感謝します。

参考文献

[1] D.R.Cheriton,"V", Communication of the ACM, Vol 31, No. 3, pp. 314-333, 1988

[2] 桜井他, 「分散環境をサポートするOS ToMの構想」, 『日経エレクトロニクス』, pp. 187-199, 1989.10.16(no.484)

[3] P.Dasgupta et al., "The Clouds Distributed Operating System", Proceedings of the 8th Conference on Distributed Computing Systems, pp. 2-9, IEEE, 1988

[4] S.J.Mullender et al., 「1990年代の分散OS, Amoeba」, 『日経エレクトロニクス』, pp. 189-205, 1990.9.3(no.508)

[5] 坂本, 「オブジェクト指向言語Objective-C」, 『bit』, Vol. 18, No. 12, pp. 12-20, 1986.11