マルチスレッドを用いた分散オブジェクトの実現方式

4H－5

武本 充治　中村 隆幸　田中 聡　久保田 稔

NTT 光ネットワークシステム研究所

1　はじめに

我々は、将来の通信システムを支える抜本的なアーキテクチャの改革のために、新しいプラットフォームを検討している [1]。本アーキテクチャでは、他のネットワーク事業者やサービス事業者との接続も視野に入れ、高い相互接続性を持つシステムの実現を目指している。我々はこのオープン化の１つとして、分散実行環境の標準である CORBA に基づいたプラットフォームを、汎用 UNIX、及び、実時間 OS 上に実装している。

通信システムのアプリケーションは、本プラットフォーム上でオブジェクトとして実行される。これらオブジェクトは、高い実行性能を獲得するために、マルチスレッド実行することが求められている。本稿では、通信システムに適したサーバ側のオブジェクトのマルチスレッドによる実現方法について検討した結果を示す。

2　アプリケーション

通信システムのアプリケーションは以下のような特徴を持つ。(1) オブジェクトの１つのオペレーションの実行時間は比較的短い、(2) 複数のオブジェクトの相互作用により１つのサービスを実現する (図 1)、(3) 高い多重性を持つ [2]。このような特徴を持つオブジェクトを UNIX や実時間 OS 上で動作させるには、プロセスより軽量であるスレッドによる実現が求められる。また、アプリケーションをマルチスレッドで実行する場合には、排他制御を考慮する必要がある。

3　オブジェクトのマルチスレッド化

常にマルチスレッド実行を行なうプラットフォームの場合、マルチスレッド実行が不要なアプリケーションを効率良く実行することができない。つまり、マルチスレッド実行を行なわない指定もできることが必要となる。

アプリケーションプログラマに、スレッド割り付けの処理をすべて任せるというアプローチ (例: プロセスフィルタ) では、その処理の指定方法が複雑であるということと、多重度を上げ、多数のメッセージを同時に対処するには実行コストが高いという問題がある。したがって、処理要求にしたがって、プラットフォームがスレッドを割り付ける方式が有効である。

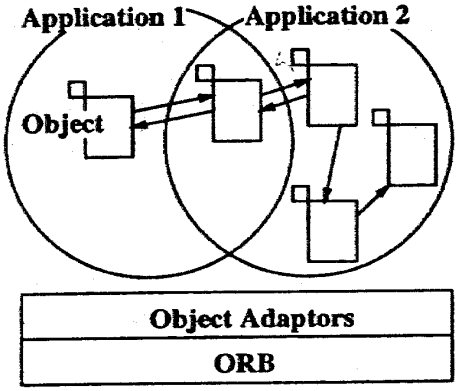

図 1: 複数のオブジェクトによるアプリケーション

3.1　スレッド割り付けの単位と排他制御

ORB 上に OA (Object Adaptor) が存在し、外部からのリクエストに応じて、この OA が該当オブジェクトインプリメンテーション (インスタンス) の該当オペレーション (メソッド) を呼び出す。ORB 上のオブジェクトの実行の形態を考えると、スレッド割りつけの単位は以下のように分類できる。

案１ OA毎: スレッドを割り付ける単位をオブジェクトを搭載・実行する OA とする方式である。OA でグループ化されたオブジェクトの中では、シングルスレッドで処理が進むために、アプリケーションプログラマは排他制御のコードは特に記述する必要はない。しかし、グループ化されたオブジェクトの内、１つでも実行中であれば、他のオブジェクトからのリクエストの処理をブロックすることになる。

案２ オブジェクトインプリメンテーション毎: 各オブジェクトインプリメンテーションには同時に１つまでしか、実行スレッドが割り付けられない方式である。この方式であれば、オブジェクトインプリメンテーションの中では、シングルスレッド実行が保証されるために、アプリケーションの高い記述性が獲得できる。しかしながら、あるオペレーション中に別のオブジェクトへ要求を出したことにより、そのオブジェクトが待たされることがあり得る。このよ

A Management Mechanism for Distributed Objects using Multi-thread Facilities

Michiharu Takemoto, Takayuki Nakamura, Satoshi Tanaka, and Minoru Kubota
NTT Optical Network Systems Laboratories
3-9-11 Midori-cho, Musashino-city, Tokyo, 180, JAPAN
takemoto@exa.onlab.ntt.jp

うな場合、同じオブジェクトへ別のリクエストが来ても、処理ができなくなる。

案3 オペレーション毎: オブジェクトインプリメンテーションの各オペレーションが並列実行の単位であるので、排他制御は、オペレーション間で共有されるデータに関する部分のみとなる。多くの場合、対象となるデータはオブジェクトのメンバ変数のみであり、アプリケーションプログラマの責任において排他制御を行なうコードを付加することになる。

案4 リクエストメッセージ毎: 複数リクエストが1つのオブジェクトインプリメンテーションの1つのオペレーションに到着しても、処理を開始できる。この場合、排他制御に関しては、上述のオペレーション毎と同様となる。

案1、案2は要求される性能から採用は困難である。スレッドの割り付け方法は、案3か案4が有力である。これら2方式は同じ排他制御、つまり、Lock & Unlockなどのコードを必要とする。異なる点は、案3が実行中のオペレーションの管理を行なう必要があるということと、案4が多重度がより高くなる可能性があることである。通信システムには、案4の方式が適している。

3.2 スレッド割り付けのタイミング

3.1で示した方式のマルチスレッド実行を行なうプラットフォームを開発中である。我々の採用するスレッドの割り付けのタイミングについて述べる。プラットフォーム上でのオブジェクトは、(1)リクエストメッセージの受信、(2)該当オブジェクトの決定、(3)オブジェクトの引数の設定、(4)オブジェクトの呼び出し、(5)返答処理、により実行される。このいずれの段階でスレッドを割り付けるかで、以下のような分類ができる。

案1 図2aに示すように、(1)でのメッセージ受信後、スレッドを割り付け、以降の処理を行なう方式。案2と比較して、早いタイミングで次のリクエストメッセージを処理できる利点がある。この方式の場合、マルチスレッド実行の指定が、OA単位になる。

案2 図2bに示すように、(2)での該当オブジェクト決定後、スレッドを割り付け、以降の処理を行なう方式。案1と比較すると、次のリクエストメッセージの処理が遅れるという欠点がある。しかし、マルチスレッドの実行の指定が、オブジェクト毎に行なえるという利点がある。さらに、スレッド生成数などで、オブジェクトの実行の可否を決定した後にスレッドを割り付けることになるので、実行効率も上げることができる。本方式が、通信システムに適切な方式である。

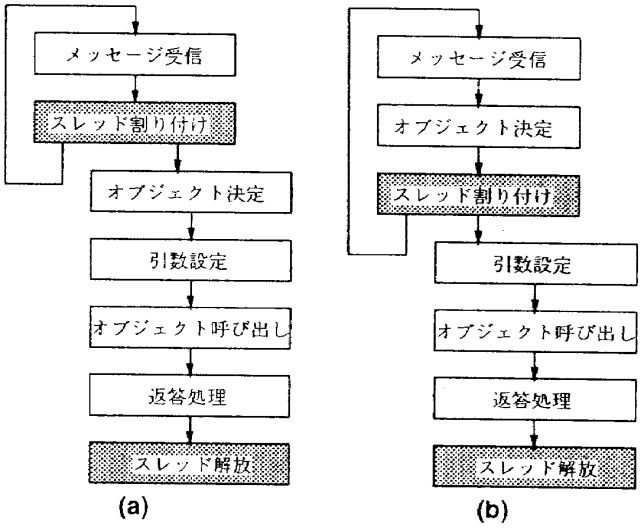

図2: スレッド生成タイミング

4 おわりに

分散オブジェクトで実現される通信アプリケーションを想定して、オブジェクトのマルチスレッド化の検討をした。その結果、リクエスト毎にスレッドを生成し、リクエストメッセージの該当オブジェクトが決定後にスレッドを割り付ける方法が、通信システムに適していることを示した。今後はスレッドのプール化などのスレッドの管理の最適な方法を決定し、本格的なプラットフォームの実装を行なう予定である。なお、現状ではSUN Solaris 2.5.1上とPLATINA[3]上のプラットフォームと、開発用のIDLコンパイラが動作している。

参考文献

[1] Suzuki, S. et al., "DONA: Distributed Object-Oriented Network Architecture for Revolutionary Network Reconstruction," in *ISS*, vol. II, pp. 459–465, 1997.

[2] 小山 稔, ほか, "新ノードソフトウェア基盤技術," *NTT R & D*, pp. 535–542, June 1996.

[3] 久保田 稔, ほか, "通信網のための分散処理プラットフォーム," 信学論 B-I, pp. 301–309, Mar. 1996.

[4] Laukien, M. et al., *OmniBroker Version 2.0.2*, Dec. 1997. http://www.ooc.com/.

[5] Lo, S.-L., *The omniORB2 User's Guide*, Mar. 1997. http://www.orl.co.uk/omniORB/omniORB.html.

[6] IONA Technologies PLC, *Orbix Programmer's Guide*, Oct. 1997.

[7] Visigenic Software, Inc., *VisiBroker for C++ Programmer's Guide Version 3.0*, 1997.