CORBAネーミングサービスの実現方式と応答性に関する一考察

2N－6

大谷 治之、楠 和浩、下間 芳樹
三菱電機（株）情報技術総合研究所

1．はじめに

近年、OMG CORBA[1]を代表とする分散システム上のコンポネント（以下、分散コンポネント）を構築するための技術が普及しつつある。コンポネントとは、再利用可能なソフトウェア部品である。従来から、言語に固有ではあるが、再利用のためのローカルなコンポネント（以下、言語コンポネント）が提供されている。C++言語に固有のコンポネントとしては、Microsoft MFC* や ANSI/ISO 標準 STL[2]が代表的なものである。

分散コンポネントを言語コンポネントを用いて実装し提供することは、アプリケーション開発の効率化の観点から非常に有用である。本稿では、STLを用いて、CORBA ORB上のコンポネントであるCORBAネーミングサービスを実際に実現し、その有効性と問題点について考察する。

2．分散コンポネントと言語コンポネント

CORBA ORB上のコンポネントを言語コンポネントであるSTLを用いて実装する、というのは自然な要求である。STLを用いることで、アプリケーションの品質と移植性を良くすることができ、開発コード量を削減できる。

分散システム上のコンポネントは、ローカルに用いられる言語コンポネントとは異なり、それを利用する複数のクライアントからの同時アクセスを考慮する必要がある。例えば、コンポネントが状態を持つ場合は、状態の一貫性を保つために排他制御を行う必要がある。

STLを使用する場合には、同時アクセスに対するデータの保護を開発者が行う必要がある。つまり、ロック保持はメソッド単位となる。メソッド単位での排他制御は、ロック保持の時間が長く、クライアントに対する十分な並列性を提供できない可能性がある。次章では、STLを用いてCORBAネーミングサービスを3つのロック方式で実現し、並列性に対する影響について検証する。

A Study of Performance Evaluation on a STL-based CORBA Naming Service
Haruyuki Ohtani, Kazuhiro Kusunoki, Yoshiki Shimotsuma, Information Technology R & D Center, MITSUBISHI ELECTRIC CORP. 5-1-1 Ofuna, Kamakura, Kanagawa 247-0056, Japan

3．STLによるCORBAネーミングサービス

CORBA ネーミングサービスは階層的な名前空間を提供し木構造を形成する（図 1）。中間のノードは、ネーミングコンテキストと呼ばれ、末端のノードはネームバインディングと呼ばれる。

我々は、ネーミングコンテキストおよびネームバインディングのそれぞれを STL マップとして実装した。STLマップは平衡二分木によるコンテナである。また、下位の CORBA ORB はスレッドプールによって動作するORBである。CORBA クライアントから送られた要求は、あらかじめ作成された複数スレッドのうち、空いているスレッドによって処理される。各スレッドは、要求・応答のマーシャリング処理とSTLマップに対するメソッドコールを行う。

検証では、合計 10 の CORBA クライアントを使い、それらが連続的にCORBAネーミングサービスに対する要求を出す。要求には、ネームバインディングをSTLマップに書き込むbind要求と、名前によってSTLマップを検索するresolve要求がある。bind要求では、STLマップに書き込みを行うと同時に、データを不揮発なものにするためにI/O処理を行う。今回はSleepコール(100ms)によってI/O処理を模擬した。次の3つの場合について、bind要求の割合を変更し、応答時間を測定した。

場合1：スレッドプール内のスレッドが1つ

この場合はSTLマップにアクセスするのは常に1つのスレッドであるため、処理をシリアラ

イズするための排他制御は行わない。

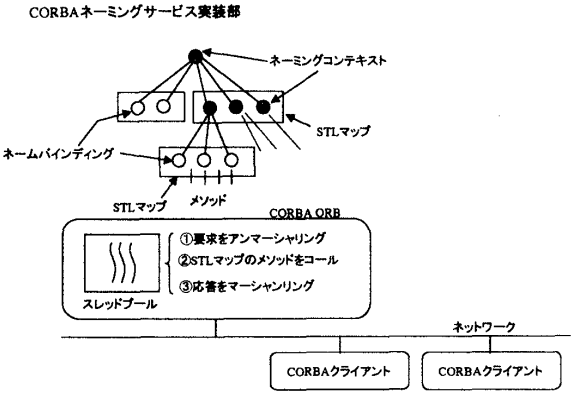


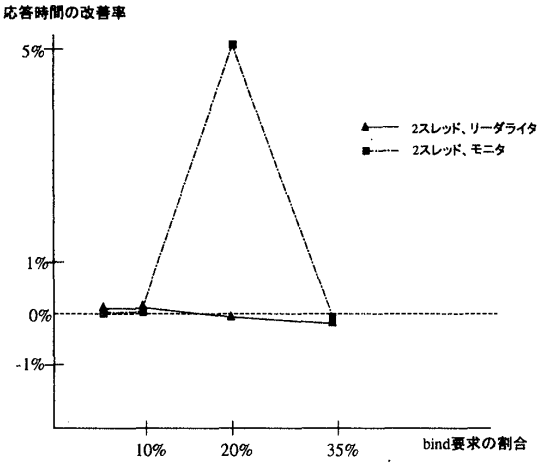
図1：STLによるネーミングサービス

場合2：モニタ
スレッドプール内のスレッドは複数あるが、STLマップにアクセスする時点で排他制御によって処理をシリアライズする。
場合3：リーダライタ
スレッドプール内のスレッドは複数あり、STLマップを検索する複数の resolve 要求は並列に処理される。但し、あるスレッドが bind 要求の処理を行うと、他のスレッドはすべて待たされる。

図2は場合1の応答時間を基準として、場合2および場合3において、どの程度応答時間が改善されるかについて示したものである。その結果、bind 要求が 10%以下の場合、リーダライタが最も応答性能が良いが、応答時間は1スレッドの場合に比べて 0.1%程度しか改善されない。bind 要求が 20%の場合、場合2のモニタが約5%応答時間を改善する。bind 要求が 35%になると、場合2および場合3ともに、場合1に比べて応答時間が 0.1%程度悪くなる。

4．考察

STLによる今回の実装では1つのSTLマップに 35%もの bind 処理が集中すると、まったく並列性が得られないことが分かった。書き込みによる長期のロック保持が、ロック競合を激しく増加させている。しかし、20%程度の bind 要求ならば、モニタによってある程度の並列性が得られる。これは、比較的 CPU 負荷の高いマーシャリング処理が並列に行えることに起因する。一方、リーダライタは期待しているほど並列性に対して効果を上げていない。resolve 処理が I/O 処理など長いオペレーションを含まないことが原因と考えられる。bind 要求が 10%以下の場合には、1 スレッドで十分に高速な処理ができるため、モニタおよびリーダライタによる並列性の効果はそれほどない。



構成：P-90 4CPUサーバ、P-200 1CPUクライアント、P-MMX 1CPUクライアント

図2: 応答時間の改善率

5．おわりに

CORBA ネーミングサービスを例に STL を分散コンポネントの構築に用いた場合の処理の並列性について検証した。本実装によるネーミングサービスについて述べるならば、名前空間を階層化し、複数の STL マップを用いれば、1つの STL マップに bind 要求が集中しないようすることができる。また、アクセスする STL マップが異なればロックの競合もそれだけ減少する。従って、実際には、モニタによる実装でかなりの並列性が提供できると予想できる。名前空間の多階層化と応答性能の関係については、今後、調査する予定である。

参考文献

[1] The Common Object Request Broker: Architecture and Specification Revision 2.0 July 1995
[2] STL Tutorial and Reference Guide – C++ Programming with the standard template library, David R. Musser and Atul Saini, Modena Software, Inc

*MFC は米国 Micorosoft 社の登録商標です