6U－4

C++のオブジェクトをプロセス間で共有するための管理機構について*

西口直樹† 園田俊浩† 柴田清己† 本田文雄† 竹林 知善†
富士通研究所‡

1 はじめに

C++によるプログラミングでは、プロセス間でデータを共有する場合、共有データの構造を決め、個々のプロセスで共有データとC++のオブジェクトとの変換を行うことが多い。また、そこでの共有データとC++オブジェクトの内容は殆んど同じであることが多い。そこで、我々は、共有データとしてC++オブジェクトそのものを使用することを提案する。

2 C++のオブジェクト

C++のオブジェクトは、メンバ変数と仮想関数のアドレステーブル(virtual table)とで構成される。仮想テーブルには、仮想関数のアドレスが入っている。また、メンバ変数には、オブジェクトへのポインタを持たせることができる。そのため、C++のオブジェクトをプロセス間で共有するためには、以下の条件を満たす必要がある。

- オブジェクトはどのプロセスでも同じ仮想アドレスに存在すること
- 仮想関数はどのプロセスでも同じ仮想アドレスに存在すること

第一の条件については、すべてのプロセスにおいて同じ仮想アドレスに共有メモリを作成し、その領域にオブジェクトを配置することによって実現できる。

第二の条件については、共有するオブジェクトのクラスの定義は共有ライブラリとして作成し、その共有ライブラリをすべてのプロセスにおいて同じ仮想アドレスにロードすることによって実現できる。

3 オブジェクト共有の方法

3.1 前提条件

ここで、C++のオブジェクトの利用の仕方を決める。オブジェクトを共有するため、利用する側では以下のことを考える必要がある。

- オブジェクトを取得する方法
- オブジェクトを変更する単位
- オブジェクトの参照状態の通知

オブジェクトの取得は、名前からオブジェクトの配置された仮想アドレスを得ることで行う。よって、オブジェクトは名前を付けられる必要がある。

オブジェクトを変更する単位をオブジェクトとすると、関連する複数のオブジェクトを変更するとき、他のプロセスとの間でデッドロックを起こす可能性がある。そこで、オブジェクトを変更する単位は、複数のオブジェクトから成るグループとする。名前を付けるオブジェクトは特別なオブジェクトと考えられるため、名前ありオブジェクトを先頭にそのオブジェクトから辿れる名前なしのオブジェクトをグループとする。

オブジェクトの参照などをオブジェクトを管理しているサーバに通知することは、必要無いオブジェクトをメモリに残しておかないために必要なことである。オブジェクトの参照状態の通知は、オブジェクトを参照するとき、オブジェクトを指すポインタを変更するとき、オブジェクトの参照をやめるときに起こる。これらの通知を自動的に行うために、オブジェクトを指すポインタにはすべてスマートポインタを使用する。参照状態が変化するときは、スマートポインタがサーバに通知するため、特別なプログラムを記述する必要は無い。

以上より、共有オブジェクトを利用する側では、オブジェクトの取得を名前で行うこと、オブジェクトの変更をグループ単位で行うことの二点において通常のプログラミングと異なることになる。

3.2 オブジェクトのバージョン

通常、データを共有する場合は、参照、変更とも排他的に行う必要がある。しかし、オブジェクト変更中にプロセスが異常終了した場合や、変更を取り消したい場合には、共有しているデータを書き換えてしまうと不都合が起こる。そこで、オブジェクトにバージョンを設け、オブジェクトを変更中のプロセスがオブジェクトを参照中のプロセスに影響を及ぼさないようにする。

これはオブジェクトからオブジェクトへのポインタが

*On the mechanism of management for sharing C++ objects with other processes

†Naoki NISHIGUCHI, Toshihiro SONODA, Seiki SHIBATA, Fumio HONDA and Tomoyoshi TAKEBAYASHI
‡Fujitsu Laboratories Ltd.

複数必要になることを意味する。そのため、オブジェクトを指すポインタは、直接オブジェクトを指さず、ある場所を指すようにし、その場所から目的のバージョンのオブジェクトのアドレスを取得することにする。オブジェクトの変更はグループ単位であるから、オブジェクトすべてを間接的にアクセスする必要はなく、名前を持ったオブジェクトのみ間接的にアクセスすればよい。オブジェクトのバージョン毎のポインタを保持する場所をグループオブジェクトと呼ぶことにする。グループの例を図1に示す。図中、矢印はポインタを示し、角の丸い四角はグループオブジェクトを示し、網かけの円は名前を持ったオブジェクトを示し、白抜きの円は名前を持たないオブジェクトを示している。また、Oの添字は、オブジェクトの識別子とバージョン番号を示している。

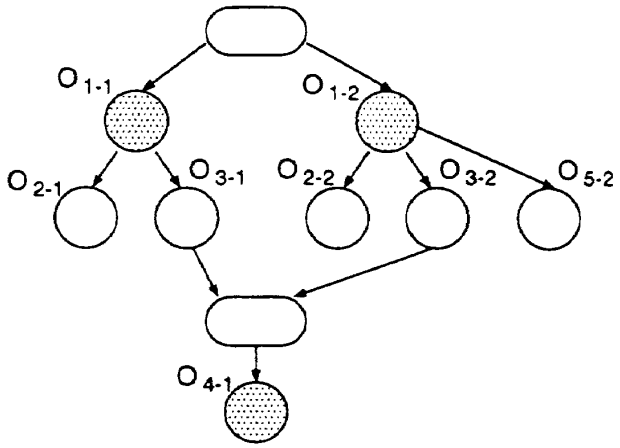

図 1: グループの例

各プロセスでは、バージョンに応じてオブジェクトをアクセスすることになるが、この処理はスマートポインタに隠蔽可能であり、プログラミングの必要はない。

3.3 共有オブジェクトの管理

3.1で述べたように、共有オブジェクトを管理するサーバの基本動作は、各プロセスと通信し、オブジェクトの参照数を増減し、参照のなくなったオブジェクトを削除することである。

しかし、3.2で述べたように、オブジェクトの変更を新しいバージョンを作成して行うことにより、サーバにおいてバージョンに関する管理を行う必要がある。

サーバは、新しいバージョンのオブジェクトの情報を必要とする。そのため、グループの変更による新しいバージョンの作成はサーバにおいて行う。このとき、同じグループの変更を一つに限定することによってグループの変更を排他的に行うことができる。

グループの変更によって、複数バージョンのグループが存在するようになり、グループを変更した以外のプロセスは、新しいバージョンを参照する方法が必要である。そのため、サーバは、グループの変更の終了時に、そのグループを参照しているプロセスに新しいバージョンが利用可能なことを通知する。通知されたプロセスは、変更されたグループのバージョンを受け、そのバージョンを参照するかサーバに伝える。サーバは、プロセスの参照するバージョンを変更し、参照されなくなったバージョンは、メモリ上から削除することができる。

以上より、プロセスがグループを変更する時は、サーバに変更開始を伝えることと、変更が終了したらサーバに変更終了を伝えることが必要になる。また、グループの変更を通知されたプロセスは、新しいバージョンを参照するか判断し、バージョンの変更をサーバに伝える必要がある。これらの処理の記述はプログラマに強いることになる。

4 実装と評価

実装は、Linux 上で GNU C++コンパイラを使用して行った。上記の仕様に基づいて、共有ライブラリを同じ仮想アドレスへロードするダイナミックリンカと共有ライブラリサーバ、共有メモリにおけるオブジェクトの管理を行うオブジェクトサーバ、個々のプロセスにおいてオブジェクトサーバとやりとりを行うローカルマネージャを作成しC++のオブジェクトの共有を実現した。

現在の仕様ではオブジェクトの変更を頻繁に行う用途には向かないが、オブジェクトの変更中も他プロセスでオブジェクトの参照を行う用途には有用である。

5 おわりに

C++のオブジェクトをプロセス間で共有する方法を検討し、それに基づいた実装を行った。共有によるプログラム記述の増加をオブジェクト変更時のみにとどめていること、オブジェクト変更時に他のプロセスを保護することを特徴としている。今後、オブジェクトサーバとローカルサーバ間の通信の効率化や、オブジェクト変更時のオーバヘッドの軽減を行う予定である。

参考文献

[1] S. K. Shrivastava, G. N. Dixon and G. D. Parrington, "An Overview of the Arjuna Distributed Programming System," *IEEE Software*, January 1991.

[2] V. Singhal, S. V. Kakkad and P. R. Wilson, "Texas: An Efficient, Portable Persistent Store," *Proceedings of the Fifth International Workshop on Persistent Object Systems*, pp. 11-33, September 1992.

[3] E. Casais, M. Ranft, B. Schiefer, D. Theobald and W. Zimmer, "OBST — An Overview," *Tech.Rep.FZI.039.1*, June 1992.