2G−1

オブジェクトの堆積に適したＲＰＣスタブ生成器

新城　靖，清木　康
(筑波大学)

1　はじめに

分散型オペレーティング・システムの構築において、近年オブジェクト指向、あるいは、object-based（継承がないオブジェクト指向）による方式が盛んに研究されている。このようなシステムでは、システムのサービスは、サーバにより管理されたオブジェクトを通じて提供される。サーバ・クライアント間の通信プリミティブとしては、遠隔手続き呼出し（ＲＰＣ, Remote Procedure Call）が最も広く利用されている。そして多くのシステムでは、そのスタブ生成器を供えている。

従来のシステムでは、１つのサービスについて複数のサーバを実現することは、あまり行われていなかった。サーバも、単体で利用されるだけで、複数のサーバを組合せて利用することは少なかった。一方、従来のスタブ生成器では、サーバの手続きのインタフェースを中心として記述するものが大部分であった。

本稿では、オブジェクトの堆積の実現に適した、遠隔手続き呼出しのスタブ生成器について述べる。オブジェクトの堆積とは、一様なインタフェースを持つオブジェクトを積み重ねることで、それらのオブジェクトの機能を統合して利用する仕組みであり、モデル化の手法である。本稿で述べるスタブ生成器では、従来のものとは異なり、サーバではなくオブジェクトを中心にインタフェースを記述する。そして、一様なインタフェースを持つオブジェクトの記述も容易である。また、既存のインタフェースを拡張したり、あるいは逆に、制限することで、新しいインタフェースを定義することも可能になっている。

2　オブジェクトの堆積

オブジェクトの堆積（object-stacking）とは、object-basedシステムにおいて、一様なインタフェースを持つ複数のオブジェクトを積み重ねることにより、それらの機能を統合して利用する仕組みである。ここでインタフェースとは、オブジェクトが受け付けることができる遠隔手続き呼出しの手続きである。また、**オブジェクトを積み重ねる**（to stack）とは、上層のオブジェクトが下層のオブジェクトの識別子を保持し、かつ、上層のオブジェクトの機能を実現するために必ず下層のオブジェクトの機能を利用することである。

オブジェクトの堆積では、比較的単純な機能を提供するサーバを複数実現する。たとえば、キャッシング、フィルタリング、マスキング、グループ化、複製、移動のような機能について、それぞれサーバを用意する。クライアントは、それらのサーバのオブジェクトを積み重ねることで、全体として高度な機能を利用することが可能となる。

オブジェクトの堆積は、並列／分散応用プログラムを対象とした分散型オペレーティング・システムＲｅＳＣの開発において考案された仕組みである[1]。ＲｅＳＣでは、高度なファイル・サービスを提供するために、広く使われる。オブジェクトの堆積は、他のobject-basedシステムにおいても利用することが可能である。ただし、そのシステムにおいて、１つのサービスについて複数のサーバが同時に利用可能であり、かつ、それらを一様に扱える機能が提供されていなければならない。

3　インタフェース記述言語の構文と意味

オブジェクトの堆積に適したインタフェース記述言語を設計した。扱えるデータ型は、Ｃ言語、および、SunRPC[3]のインタフェース記述言語（rpcgenコマンドの入力）を拡張したものになっている。拡張した型構成子としては、proc, sproc, set of, server があげられる。インタフェース記述は、次のような文の並びである。

＜名前＞：＜型＞＝＜値＞；

図1に示した簡単なファイル・サーバの記述を例に、この言語の特徴について説明する。

型構成子procは、オブジェクト手続きを、sprocは、サーバ手続きを定義する。オブジェクト手続き／サーバ手続きとは、それぞれオブジェクト／サーバに対する遠隔手続き呼出しのエントリで、オブジェクト指向言語におけるインスタンス・メソッド／クラス・メソッドに相当するものである。たとえば、第１０行では、手続きreadが２つの整数型の値を入力パラメタとし、バッファ型（buff_t）の値を出力パラメタとするオブジェクトの手続きとして定義されている。

オブジェクト手続きは、隠れたパラメタとしてオブジェクト識別子（oid_t）を含むようなサーバ手続きである。たとえば、オブジェクト手続きreadは、次の様なサーバ手続きと等価である。

```
read: sproc( self:oid_t; where,count:int ) buff_t = 52 ;
```

オブジェクト手続き、および、サーバ手続きの変数の値（たとえば、readでは52）は、遠隔手続き呼出しの手続き番号である。この値は、全体で唯一でなければならない。スタブ生成器は、この唯一性をチェックする。

この言語の特徴は、型構成子set ofを用いてインタフェースを手続きの集合として記述できる点にある。たとえば、第７行では、any_serverという変数にサーバ手続きcreateとcopyを要素とする集合が代入されている。これは、どんなサーバもそのサーバ手続きをインタフェースに含んでいることを意味している。

An RPC Stub Generator for Object-Stacking

Yasushi SHINJO and Yasushi KIYOKI
University of Tsukuba (Email: yas@is.tsukuba.ac.jp)

```
 1: buff_t: type = array of opaque ;
 2:
 3: kill:    proc( void ) void                    = 14 ;
 4: create: sproc( lower:array of oid_t ) new:oid_t =  1 ;
 5: copy:   sproc( orig:oid_t )  new:oid_t        =  2 ;
 6:
 7: any_server: set of sproc = { copy, create };
 8: any_object: set of  proc = { kill };
 9:
10: read:  proc( where,count:int ) buff_t = 52 ;
11: write: proc( where,count:int;buff:buff_t ) void = 53 ;
12:
13: file: set of proc = any_object + { read,write };
14: read_only_file: set of proc = file - { write };
15: file_server: type = server( any_server + file );
16:
17: StdFS: file_server = 3 ;
18: ZFS  : file_server = 5 ;
19: CFS  : file_server = 6 ;
20: ReadOnlyFS : server( any_server+read_only_file ) = 7 ;
```

図1　簡単なファイル・サーバの記述（本記述言語）

この言語では、手続きの集合を扱う演算として、集合の和と集合の差がある。たとえば、第１３行では、任意のオブジェクトに対して、読み込み、および、書き込みを行う手続きを追加して、ファイル・オブジェクトのインタフェースを定義している。次の行では、読込み専用のファイル・オブジェクトのインタフェースを、普通のファイル・オブジェクトから書込みのエントリを削除することによって定義している。

型構成子serverは、手続きの集合を引数としてサーバ（サーバ識別子型）を定義する。第１５行では、ファイル・サーバを、任意のサーバとしての手続きとファイルの読み書きを行う手続きを受け付けることができるサーバとして定義している。（オブジェクト手続きは、オブジェクト識別子を隠れた引数として持つような特殊なサーバ手続きであることから、サーバ手続きの集合とオブジェクト手続きの集合の和を求めることができる。）ＲｅＳＣでは、この型の変数の値は、サーバ識別子として利用するので、全体で唯一でなければならない。スタブ生成器は、その唯一性をチェックする。

このスタブ生成器は、以下の点においてオブジェクトの堆積に適している。

(1) サーバできなくオブジェクトを中心にインタフェースを定義することが可能となっている。このことは、オブジェクトを中心に考えるオブジェクトの堆積に合致している。

(2) オブジェクトの堆積において重要となる一様なインタフェースを持つオブジェクトを容易に定義することができる。

(3) オブジェクト手続きが特殊なサーバ手続きであるというobject-basedシステムの性質を直接的に反映した記述が可能となっている。

第４行において定義されているサーバ手続きcreateがオブジェクトを積み重ねる操作を行うものである。この手続きは、引数としてオブジェクト識別子の配列を受け取り、それらを下層のオブジェクトとして新しいオブジェクトを生成する。この手続きは、全てのサーバのインタフェースに含まれていなければならない。

4　スタブ生成器の実現

現在、第１バージョンとして、この定義を読み込み、SunRPCのスタブ生成器であるrpcgenのソース・プログラムを生成するものを開発している。そのスタブ生成器は、図１のような記述を読み込み、図２に示すようなプログラムを生成する。図１において散在している手続きの記述は、図２の第２０行から開始しているプログラム（サーバのインタフェース）の記述に集約されている。rpcgenでは、入力パラメタと出力パラメタの数がそれぞれ１個に制限されているので、スタブ生成器によりダミーの型を定義する必要がある。たとえば、図１の第１０行における手続きreadの入力パラメタは、図２の第１４行で定義されている構造体read_arg_tにまとめられている。図２の第１５行のselfは、前章で述べたオブジェクト手続きの隠れたパラメタである。

```
 1: enum server_t { StdFS=3, ZFS=5, ZFS=6, ReadOnlyFS=7 };
 2:
 3: struct oid_t {
 4:         site_t          site_id;
 5:         server_t        server_id;
 6:         object_number_t object_number;
 7: };
 8:
 9: typedef opaque buff_t<>;
10:
11: struct create_arg_t {
12:         oid_t   lower<>;
13: };
14: struct read_arg_t {
15:         oid_t   self ;
16:         int     where ;
17:         int     count ;
18: };
19: ・・・
20: program FILE_SERVER_PROGRAM {
21:     version FILE_SERVER_VERSION {
22:         void    KILL(oid_t)             = 14 ;
23:         oid_t   CREATE(create_arg_t)    =  1 ;
24:         oid_t   COPY(oid_t)             =  2 ;
25:         buff_t  READ(read_arg_t)        = 55 ;
26:         void    WRITE(write_arg_t)      = 56 ;
27:     } = 1;
28: } = 600000000 ;
```

図2　簡単なファイル・サーバの記述（SunRPCのrpcgenの入力）

第１バージョンでは、オブジェクト手続きを実現していない。すなわち、オブジェクト手続きでは、オブジェクトに対して遠隔手続き呼出しを行う必要があるが、SunRPCでは、サーバに対して遠隔手続き呼出しを行うようなスタブしか得られない。現在は、それを人手でオブジェクトへの遠隔手続き呼出しに書き換えている。第２バージョンでは、この部分を自動生成するものを作成する予定である。

5　むすび

本稿では、オブジェクトの堆積とそれに適したスタブ生成器のインタフェース記述言語とその実現について述べた。今後、Ｃ＋＋のようなオブジェクト指向言語に対するスタブ生成器について研究を進めていきたい。

参考文献

[1] Y.Shinjo and Y.Kiyoki: "ReSC: A Distributed Operating System for Parallel and Distributed Applications", Proc. First Intl. Conf. on Parallel and Distributed Information Systems, p.171, Dec. 1991

[2] 新城，清木："分散型オペレーティング・システムにおけるオブジェクトの堆積"，情報処理学会 第４２回全国大会講演論文集，3K-8，第４分冊 pp.15-16，1991.

[3] "Network Progamming Guide", Sun Microsystems, Inc. 1990.