3N−2

オブジェクト共有空間を利用した分散プログラミング支援フレームワーク

前田 直人† 上田和紀‡
†早稲田大学大学院理工学研究科 ‡早稲田大学理工学部

1 はじめに

本稿ではオブジェクト共有空間を基にして作成したJava言語環境(以降Javaと略す)による分散プログラミング支援フレームワークSOR (Shared Object Repository)の構成と実装について述べる。SORはオブジェクトの共有モデルを利用して、位置透過かつ通信透過な分散システムの記述を可能にする。オブジェクトの共有を利用した分散システムのための並列言語[1]やフレームワーク[2][3]はすでに研究されているが、SORは特定の戦略や方針に従った共有空間を実装するためのクラスライブラリを提供しており、SORに独自の実装を持った共有空間を付加できるという特徴を持つ。それによってSORクライアントは分散システムの性質に応じて共有空間の実装を選択できるようになっている。共有空間内のオブジェクトを集中的に管理するのか分散して管理するのか、通信路はTCPソケットかUDPソケットかまたはnativeメソッドを利用してプラットフォーム依存な高速な通信路を利用するのか等は共有空間の方針によって異なってくる。共有空間に登録するオブジェクトにはユーザ定義の任意の型を利用することが可能である。

2 SOR (Shared Object Repository)

2.1 SORの構成

SORにおけるオブジェクト共有空間は、オブジェクトの格納、オブジェクトの削除、格納されたオブジェクトのメソッド呼び出しの機能を持つ。オブジェクト共有空間は想定する分散システムにより様々な実装が考えられる。そこでSORシステムは内部に複数のオブジェクト共有空間の存在を許している。オブジェクト共有空間の実装のためにRepositoryContextをはじめとするインタフェースと再利用が可能と考えられるいくつかのクラスを提供している。詳細は次の節で述べる。

複数のオブジェクト空間を管理したSORサーバ群によって一つのSORシステムが構成される。オブジェクト共有空間はさらにそのなかの同種のRepositoryContextの実装の集合によって構成される。同種のRepositoryContextの実装はお互いに通信しオブジェクトの共有状態に整合性と一貫性を持たせるように動作する。

2.2 RepositoryContext

RepositoryContextインタフェースはオブジェクト共有空間へのアクセスのためのメソッドput, remove, bind, unbind, lookupを提供する。これらの機能を実現するためにさらに次に述べる3つのクラスを規定している。共有空間にputすることで共有されるオブジェクトを共有オブジェクトと呼ぶこととする

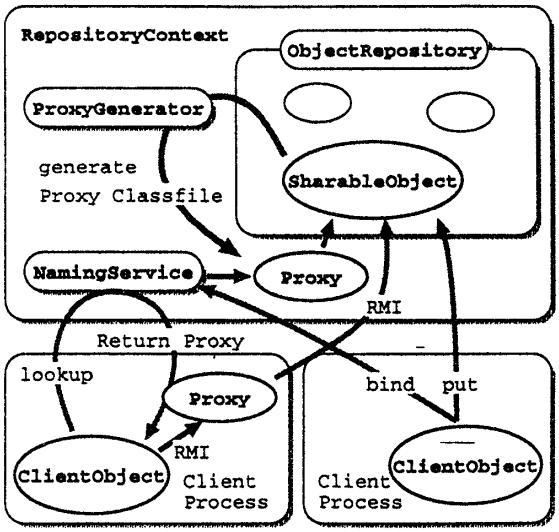

図 1: RepositoryContext概略図

ObjectRepository

共有オブジェクトをインスタンス化しそれを共有空間内にputすると、分散したホスト上にあるJVM(Java Virtual Machine)のオブジェクトから共有空間内に存在する同一のオブジェクトのメソッドを呼ぶことができるようになる。ObjectRepositoryはputされた共有オブジェクトに対してRepositoryContext内でユニークなIDを割り当て、IDと共有オブジェクトの表を管理する。Proxyはこのユニークな IDを使って対象とするオブジェクトを発見する。メソッド呼び出しや新たな参照が長い間起らずに期限が切れた共有オブジェクトは参照を切断してGCされるかもしれないし、メモリからディスクに追い出されるかも知れない。

NamingService

共有オブジェクトと文字列を関連付け、名前によって

A Distributed Programming Framework using Shared Object Space
Naoto Maeda, Kazunori Ueda.
Information & Computer Science, Waseda Univ.

共有オブジェクトを検索できるlookupのサービスを提供する。lookupは返り値としてそのProxyオブジェクトを返す。RepositoryContextがRepositoryContextを含むような階層構造である場合は関連付けられる名前も構造化する必要があるであろう。

ProxyGenrator

Javaの提供するRMI[4](Remote Method Invocation)はProxyによって通信をメソッド呼び出しに隠蔽し、通信にまつわる実装の手間を軽減してくれる。しかしRMIが参照モデルの実装として唯一提供しているUnicastRemoteObjectでは呼び出すべきオブジェクトの物理的な位置は固定されているという制約があるため、SOR内での共有オブジェクトの移動や複製の作成は困難である。従ってSORでは独自に共有オブジェクトのメソッドを呼び出すための機構を用意する。共有オブジェクトのメソッド呼び出し時にProxyによって通信を隠蔽する方法は同じであるが、通信路、引数や返り値の扱いがオブジェクトの共有の戦略によって異なるためRepositoryContextは独自にProxyGenratorを作成する必要がある。共有オブジェクト作成の手順はRMIとほぼ同様である。

3 RepositoryContextの実装

同期または非同期分散型のCSCWを目的としたマルチユーザ型の分散システムを想定したRepositoryContextの実装の具体例について紹介する。

共有オブジェクトの位置は登録されたSORサーバ上のホストに固定し、リモートメソッド呼び出しの返り値や引数の扱いはほぼRMIのそれと同一でJavaの直列化APIをそのまま利用した。lookupで使用する文字列は構造を持たない。特徴としてデザインパターンのObserverモデルのためのクラスの提供と永続オブジェクトが挙げられる。

SORにおけるメソッド呼び出しはクライアントのオブジェクトから共有オブジェクトへの一方通行であり、CallBackを行う共有オブジェクトを記述できない。そのため共有オブジェクトにたいしてObserverデザインパターンを適用することができない。Observerデザインパターンはオブジェクトの共有というモデルに関しては特に有用なパターンであると考えられるためその回避策としてRepositoryContextがObservableクラスを提供している。参考として図2に簡単なObservableカウンタの共有オブジェクトの実装を示す。

共有オブジェクトのObserverになるにはupdateメソッドを実装してProxyのaddObserver(Observer)によって自身を登録すれば良い。共有オブジェクトのnotifyObservers()が呼ばれたらそのProxyによってObserverのupdateメソッドが呼び出される仕組みになっている。

```
public class SharableCounterImpl
  extends sor.basic.ObservableImpl
  implements SharableCounter {
  int counter ;
  public SharableCounterImpl(){counter= 0 ;}
  public int inc(){
    setChanged() ;
    notifyObservers(new Integer(++counter)) ;
    return counter ;
  }
}
```

図 2: ObservableCounterの実装プログラム

NamingServiceにbindされている共有オブジェクトはProxyからの参照がなくなってもGCされない。共有オブジェクトがGCされるのはProxyからの参照がなくNamingServiceからも参照されていない場合のみである。従ってNamingServiceを利用することで永続的な共有オブジェクトを作成することが可能である。

以上の実装により同期型のチャットや共有ホワイトボードはもちろんデータを永続化することで非同期型の掲示板等も容易に記述することが可能である。

4 まとめと今後の課題

SORに登録される共有オブジェクトが常に信頼できるとは限らない。そのため信頼の置けない共有オブジェクトの動作はAppletのSandBoxのように制限するかProxyを渡す前に認証を行うなどのセキュリティのための機能をフレームワークに組み込む必要がある。

次に共有オブジェクトをローカルにcacheして高速化を目標としたRepositoryContextの実装を考えている。

参考文献

[1] Bal. H.E., Kaashoek, M.F., and Tanenbaum A.S.: "Orca: A Language for Parallel Programming of Distributed Systems", IEEE Transactions on Software Engineering, vol.18, No.3, March 1992, pp. 190-205

[2] "JavaSpaces Specification", Sun Microsystems,Inc., 1998., http://java.sun.com/products/javaspaces/

[3] "Shared Data Objects", IBM, 1998., http://www.alphaWorks.ibm.com/formula/sdo

[4] "Java RemoteMethodInvocation Specification", Sun Microsystems,Inc., 1997., http://java.sun.com/products/jdk/rmi/