基于 CORBA 的 Matlab 随机模型在计算机应用系统中的应用

Matlab Stochastic Models in Computer Application Systems Based on CORBA

权小红　(常州信息职业技术学院 软件学院 江苏 常州 213164)

摘　要：介绍一种基于 CORBA(Common Object Request Broker Architecture)接口的 Matlab 随机模型在计算机应用系统上的实现方案。应用该方案在能源交易系统上进行的期权定价模型实验结果表明,该方案有效地缩短系统运行时间,具有一定的通用性。

关键词：Matlab CORBA 随机模型 能源交易系统

1　Matlab C++编译器

作为一种功能强大的数学模型构建工具,Matlab 在矩阵计算及图形处理方面有着多数 4GL 计算机语言无法比拟的优势[1]。然而 Matlab 是一种解释性的,面向过程的语言,Matlab 模块需要进行适当的转换之后才能够集成到现有的计算机应用系统中。

Matlab 提供的 C++编译器能够将 Matlab 代码编译成动态链接库(DLL)并提供相应的 C++源代码[2]。在 Matlab 环境中运行命令 mbuild - setup 建立编译器环境后,本地的 C++编译器被搜索出来。接着运行命令:mcc -W cpplib:<libname> <Matlab .m file(s)> -T link:lib -v -d '<dirname>' 把 Matlab 的源代码编译成 DLL 并生成相应的 C++源代码及头文件。

2　CORBA C++ 接口

动态链接库(DLL)作为通用的软件组件,可以安装在本地或远程计算机上。接下来需要考虑如何实现对 Matlab 生成的 DLL 进行远程及多线程调用。通用对象请求代理架构(CORBA)为用不同语言开发的,运行在不同计算机系统上的软件组件之间的通讯提供了一个通用标准[4]。

CORBA 的系统框架如图 1。

CORBA 使用接口定义语言(IDL)来定义对外公布的接口。IDL 是语言无关的,可以在不同操作系统上用不同语言实现。典型的计算机语言如 C,C++[3],JAVA 等都可用来实现 CORBA 接口。

在 CORBA 框架中,应用程序之间的通讯通过对象请求代理(ORB)实现。在实际应用中,ORB 被初始化后与内部的对象适配器联接。对象适配器则负责诸如引用计数、对象实例化、对象生命周期管理等工作。对象适配器的另一项重要工作是在用户 IDL 代码被编译成操作系统可识别的生成代码类之后,注册生成代码类。不同的计算机语言在实现 CORBA 的方式上有所不同。其中 IDL-JAVA 十分简单明了,可方便地从 JAVA 应用程序中实现对 CORBA 接口的调用。而 C++的实现方式比较复杂,几乎包括了所有 CORBA 定义的内容[5]。

以下例子演示一个用 C++ 实现的客户端对远程组件的调用:

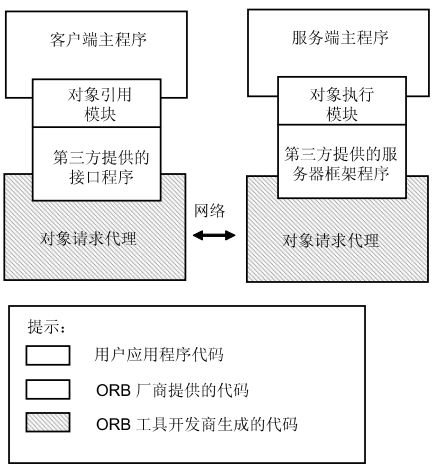

图 1　CORBA 的系统框架

首先定义一个简单的接口：

```
interface Hello
{
    void hello( ) ;
    } ;
```

然后使用 IDL 编译器编译该接口并生成 C + + 文件。组件安装并运行在服务端，在初始化后通知 ORB 以接收客户端发送的请求。以下列出最基本的服务端代码：

```
int main( int argc, char * argv[ ], char * [ ])
{
    try
      {
      // 创建 ORB 及 BOA 对象
      CORBA_ORB_var orb = …
        CORBA_ORB_init( argc, argv) ;
      CORBA_BOA_var boa = …
        orb - > BOA_init( argc, argv) ;
    // 实例化执行对象
    Hello_var p = new Hello_impl ;
    // 存储对象引用
    CORBA_String_var s = …
      orb - > object_to_string( p) ;
    const char * refFile = " Hello. ref" ;
    ofstream out( refFile) ;
    if ( out. fail( ) ) return 1 ;
    out. close( ) ;
    // 初始化完成，服务端准备接受请求
    boa - > impl_is_ready( …
      CORBA_ImplementationDef : : _nil( ) ) ;
    }
    catch ( CORBA_SystemException& ex)
    {
      OBPrintException( ex) ;
      return 1 ;
    }
    return 0 ;
    }
```

以下代码演示客户端如何远程调用实例化的服务端代码：

```
try
  {
    // 创建 ORB 对象
    CORBA_ORB_var orb = …
      CORBA_ORB_init( argc, argv) ;
    // 获取执行对象指针
     const char * refFile = " Hello. ref" ;
     ifstream in ; in. open( refFile) ;
     if( in. fail( ) ) return 1 ;
     char s[ 1000 ] ; in > > s ;
    // 创建 CORBA 及执行实例对象
    CORBA_Object_var obj = …
        orb - > string_to_object( s) ;
    Hello_var hello = Hello : : _narrow( obj) ;
    //… 具体的程序执行代码…
    }
    catch ( CORBA_SystemException& ex)
    {
     OBPrintException( ex) ;
     return 1 ;
     }
```

3 随机模型

随机模型作为一种估计概率分布的工具，其输入端变量是随机变化的。这些随机变量通常是可观察的某一阶段的历史记录。随机模型的输出变量所遵循的概率分布则可以通过大批量随机模拟推导出来。随机模型最早应用在物理学中，也被称作 Monte Carlo（MC）方法。如今随机模型在工程学，生命科学以及金融领域得到了广泛的应用。

随机模型通常使用数值方法进行数量估计。在其它分析方法无法获得精确解的情况下，随机模拟方法的优势是十分明显的。然而这种方法的缺点是它依赖大批量的模拟计算，对计算机的软硬件要求较高。为了使 MC 模拟的误差减少一个数量级，所需的 MC 迭代次数必须提高两个数量级。相应地，系统的运行时间也要大幅度增加。这就是我们之所以选择利用 Matlab 而不是其它 4GL 语言构建随机模型的原因。Matlab 的矩阵计算功能十分强大，正好用来处理海量数据。

4 应用实例

ZaiNet 是由美国 SunGard 公司开发的能源交易系统。该系统用来对能源类商品及其金融衍生物进行管理。除了支持实时的能源交易之外，该系统的另一项重要功能是提供诸如能源类商品的市场标价，收益分析和风险管理[6]。在通常情况下，ZaiNet 内建的统计分析模型可以满足以上分析之需要。然而能源商品在经历了市场解构之后变得越来越复杂，在有些情况下必须使用复杂的随机模型进行上述分析。新版本的 ZaiNet 提供了一个称为外部模型（TXM）的模块以实现系统与用户自定义模型的无缝联接。当 ZaiNet 的客户端运行统计分析时，系统检查该分析是否需要借助外部模型，如果需要则将分析请求转给 TXM 模块。

TXM 模块对 ZaiNet 客户端的请求及分析结果的反馈是通过 CORBA 实现的。ZaiNet 使用 GNU 组织认证的 omniORB，以 C ++ 实现了基于 CORBA 的 ORB。在该框架下，我们首先在 Matlab 上开发了一个期权定价模型。该随机模型使用 MC 方法模拟出期权标的物在到期日的价格概率分布，然后折算出目前的期权价格。程序在 Matlab 环境中开发调试完成后，我们使用前面介绍的方法把 Matlab 模型代码编译成 DLL 并发布在 TXM 服务器上。

为了接受来自 ZAINET 客户断的请求，还需要在 TXM 服务器上使用 omniORB 提供的 C ++ 接口编写一个包裹程序。在该包裹程序中，首先要初始化一个 ORB 对象，然后使用称作类工厂的类对封装在每个 DLL 中的模型对象进行初始化。以上面提到的期权定价模型为例，当 ZAINET 客户端的请求经由 TXM 发送给包裹程序后，类工厂创建并维护一个期权定价模型的线程实例。在模型运行并将分析结果返回给 TXM 后，模型实例被摧毁并释放资源空间。

5 比较分析

为了测试本解决方案的实际效果，我们同时开发了一个纯粹的 C ++ 应用程序。在此程序中，我们遵循相同的建模方式，在同一个硬件平台上实现对同一期权定价模型的 MC 模拟。下表比较了不同的 MC 迭代次下两种解决方案以秒为单位的运行时间开销：

表 1　两种解决方案的运行时间比较

MC 迭代次数	C ++ 应用程序	Zainet TXM 框架
1,000	48.05	45.22
10,000	153.35	78.24
100,000	1382.47	296.66
1,000,000	13505.02	1076.54

从表中我们清楚地看到，随着迭代次数的增加，本文所提供的解决方案在时间开销上的优势是明显的。在百万级的迭代次数下，本解决方案在系统运行时间上缩短了一个数量级。这在对系统的实时性及多用户并发请求要求较高的系统中是非常关键的提高。

6 结束语

随机模型在金融等领域的广泛应用需要一种有效的解决方案，以使得快速高效的模型开发能够和传统的应用系统有机地结合起来。本文所提供的方案旨在解决这一问题。从实际运行效果来看，基于 CORBA 的 Matlab 随机模型无论在系统运行速度上还是在标准化程度上，较之传统的解决方案都有明显的优势，适合在一定范围内推广应用。

参考文献

1 Gilat A. MATLAB: An Introduction with Applications 2nd Edition. John Wiley & Sons. ISBN 978 - 0 - 471 - 69420 - 5,2004.

2 QuarteroniA, Saleri F. Scientific Computing with MATLAB and Octave. Springer. ISBN 978 - 3 - 540 - 32612 - 0,2006.

3 苏兹斯基. 王千祥等译. 构件化软件：超越面向对象编程（第二版）. 北京：电子工业出版社,2004.

4 Mowbray TJ, Zahavi R: The Essential Corba: System Integration Using Distributed Objects, John Wiley & Sons, ISBN 0 - 471 - 10611 - 9.

5 Henning M, Vinoski S. Advanced CORBA Programming with C ++, Addison - Wesley, ISBN 0 - 201 - 37927 - 9.

6 约翰·赫尔. 张陶伟译. 期权·期货和其它衍生产品. 第三版，北京：华夏出版社,2000.