面向方面在实时系统中间件中的应用

刘东星,张立臣,高 娜
(广东工业大学 计算机学院,广东 广州 510006)

摘 要:实时系统中间件存在着分布性、实时性、容错性、安全控制、性能分析、日志记录等非功能的横切关注点,基于面向方面的中间件构件技术就是要把这些非功能的要求从中间件的核心功能中分离出,形成非功能方面,运用面向方面的编程技术可实现中间件核心功能关注点和非功能的横切关注点的并行设计与开发,这增加了中间件设计的模块性、可扩展性和可维护性。文中提出了采用面向方面的编程思想,重构基于 Java 的 RMI(Remote Method Invocation)分布式框架,实现实时系统中间件的分布性。

关键词:实时系统中间件;面向方面;RMI
中图分类号:TP311.52 **文献标识码**:A **文章编号**:1673-629X(2010)07-0040-04

Application of Real-Time System Middleware Based on Aspect Oriented

LIU Dong-xing, ZHANG Li-chen, GAO Na
(College of Computer, Guangdong University of Technology, Guangzhou 510006, China)

Abstract: Aspect oriented middleware can separate non-functional crosscutting concerns, such as distribution, real-time, fault-tolerance, security, performance analysis, logging, etc., from the core functions of middleware, and implement them as non-functional aspects, using aspect oriented programming the crosscutting and core function concerns can be designed and developed concurrently and independently. Aspect oriented software development provides middleware more modularity, expansibility and maintenance. In this paper, used aspect oriented programming model, distributed by RMI(Remote Method Invocation) framework based on Java, distribution of real-time system middleware has been designed.

Key words: real-time system middleware; aspect-oriented; RMI

0 引 言

按照 IDG 的定义,中间件是一种独立的系统软件或服务程序,分布式应用软件借助这种软件在不同的技术之间共享资源,中间件位于基于客户/服务器的操作系统之上,管理计算资源和网络通信。目前有相当多的企业已经充分应用了 CORBA, J2EE, .NET, Web Services 等中间件来解决其复杂的分布式计算问题,如事务处理、消息传递、数据通信、安全性、性能监控[1]。在中间件系统开发的多个需求中,包含两类需求,即核心模块级需求和系统级需求。系统级需求通常是相互独立的并且会横切核心模块级需求。

对于一个分布式程序开发者来说,首先希望自己将精力注重在核心模块功能上[2]。然后,加入支持分布式应用的代码。但在面向对象的软件构造方法中,分布式应用的代码作为系统级需求和核心模块级需求的功能代码是紧密耦合的,它会带来二个方面的影响,一方面是代码混乱:软件系统中的核心模块可能要同时兼顾几个方面的需要,开发者经常要同时考虑业务逻辑、分布式通信等问题,兼顾各方面的需要引起核心模块代码的混乱。另一方面是代码分散:由于分布性通信涉及到多个核心模块,相关实现也得遍布在这些模块,代码的分散降低了系统的性能。整个系统在开发期提高了开发难度和复杂度,增加了测试难度,系统可维护性和可扩展性差。采用面向方面的编程思想,可以自然地分离和模块化系统的功能性需求和非功能性核心需求,把横切系统非功能关注点形成方面,然后通过方面纺织器将方面和类整合,整合后的代码能根

收稿日期:2009-11-23;**修回日期**:2010-02-21
基金项目:国家自然科学基金重大研究计划(90818008);国家自然科学基金项目(60774095,60474072Z);广东省自然科学基金项目(07001774,04009465)
作者简介:刘东星(1983-),男,硕士研究生,研究方向为软件设计与实时系统;张立臣,教授,博士,研究方向为并行处理、分布式处理、实时系统。

据具体应用环境自适应剪裁后编译成可执行代码。

文中主要是针对运用 RMI 分布式框架进行通信的中间件,采用面向方面的编程思想,重构此类中间件的分布式框架。通过对中间件的分布式关注点进行方面的分解、关注点的实现、方面的织入,由方面类实现运行时功能性代码跟分布性的自动整合,这样在整个系统的开发过程中可以将分布式通信单独形成模块,系统的开发环境就像在单机中的开发环境。

1　面向方面编程(Aspect - Oriented Programming)方法

1.1　面向方面的基本思想及开发步骤

在软件开发方法的演化过程中,面向过程与面向对象对现实世界需求的建模都采用纵向开发方法,现在已广泛应用的面向对象技术通过对对象的抽取,特别是应用设计模式对逻辑混杂问题的处理,分解了模块间的耦合性,同时加强了模块的内聚性。但是无法处理对于横切多个模块的非功能模块[3]。面向对象是对现实世界的处理,所用的分离关注方法对功能性的关注进行了相当好的抽象和建模,相对于面向过程,面向对象更加理顺了抽象与细节之间的关系[4]。非功能系统关注横切了其它功能性核心模块,这就涉及到了多纬抽象的考虑,但是面向对象只能处理一纬抽象。所以必须应用多纬分析法来分解此类关注点[3]。面向方面编程方法就是应用多纬的处理方法来解决非功能的横切关注点,方面模切类,打破了类的封闭性,这是一种开放的编程方法[3]。面向方面编程是一种基于关注分离的新技术,可以各自独立地分离并设计不同的关注点,特别是解决面向对象中横切其它模块的非功能关注点[5]。例如传统的结构化程序设计和面向对象编程方法中,非功能关注点的模块分散在系统中,给系统的开发和维护带来极大的难度,应用面向方面编程方法不仅能解决这个问题,而且能提高每个模块的重用性[6]。

在面向方面软件设计方法中,系统的建模主要有核心组件和方面这两部分组成。系统的核心功能性需求中包含的一系列非功能性关注点,如实时性、安全性、日志、同步控制、调度、性能优化、通信管理、资源共享、分布式管理、错误和异常处理,应用面向方面的建模技术,系统开发者可以在系统设计时从核心功能性需求中分离出这些非功能性关注点[7]。然后,系统的集成通过方面的组合和绑定来实现[5]。在系统的设计中通过分离关注点,开发者只要实现所需的方面,不用考虑其它方面[5]。

系统被方面分解机制分解为二部分:系统核心功能和贯穿系统的需求即横切关注点。可以通过面向对象或结构化方法实现系统核心功能的编码,利用面向方面的语言支持横切关注点的开发形成方面代码,独立开发这两个过程,最后通过面向方面语言的联结机制将方面代码织入到核心功能代码中,形成最终的系统代码[8]。

面向方面的三个开发步骤[8]:

(1)方面分解:按照系统的需求,分离出系统核心模块级关注点和横切关注点。

(2)关注点实现:并行开发实现不同的关注点。

(3)方面的重新组合:通过方面集成器所创建的模块单元来指定重组的规则,织入就是使用这些信息来建构最终的系统。

1.2　文中应用到的面向方面概念

横切不仅是一种对系统进行专门分解的功能,也是一种实现基本支撑环境的功能。横切有两种类型:动态横切和静态横切。动态横切是通过切入点和连接点在一个方面中创建行为的过程,连接点可以在执行时横向地应用于现有的对象[9]。

方面(Aspect)是用于实现某个非功能核心关注点的模块单元,它扩展了类的概念,通过了模块化方式实现横切关注点。

连接点(Join Point)是用于组件代码可以作用到的点,在程序执行过程中的某个特定的点,一个连接点总是代表一个方法的执行,连接点是一个抽象的概念。

通知(Advice)是一种在连接点执行的行为,当到达由切入点表达式指定的连接点时,通知指定要执行的代码。

切入点(Pointcut)是由切入点正则表达式描述的一系列连接点的组成,匹配连接点的断连。

织入(Weaving)是把切面连接到其他的应用程序类型或者对象上,并创建一个被通知的对象。

2　系统实现框架

RMI 是一个分布式对象模式,它提供了一种远程的方法调用。远程方法调用的原理是利用在不同的计算机间互相调用对方函数,然后启动对方进程的一种机制,在这种机制的协调下,当本地机上某对象调用远程计算机方法时,可以用本地机上的调用语法规则来代替对远程计算机的操作。这种调用简单方便,可以传递复杂 Java 对象。现在采用 J2EE 为主流技术的通用型中间件,J2EE 中的 EJB 的底层实现技术就是 RMI,EJB 的调用就是经过封装的、更高级的 RMI 调用。

RMI 使得使用 Java 开发分布式程序更加容易。

由于不需要设计协议(这基本是一个错误的任务)使得使用 RMI 开发分布式程序比使用 socket 更加容易。在 RMI 里面设计者就象在调用一个本地的类的方法一样,而实际上是在调用的时候相应的参数被发送到远端的对象和然后被解释。最后结果返回给调用者。

在 RMI 远程方法调用中,客户通过调用位于客户端的 Stub 对象中的方法与远程分布式服务器透明的进行通信[10]。

面向方面的编程思想主要要解决的问题就是划分商业逻辑代码中存在的日志记录、性能统计、安全控制等,然后把这些非功能性需求看作为系统单独所要解决的问题[11]。

基于 Java 的 RMI(Remote Method Invocation)分布式框架进行通信的实时系统中间件,其通信接口分为如图 1 所示的四个层次,分别为业务访问层、RMI 通信层、业务调用层、业务配置层[12]。

(1)业务访问层的主要作用是为客户端提供访问接口,客户端可以通过这个业务访问层访问服务端的资源,对于客户端来说通信是透明的[13]。

(2)RMI 通信层的主要作用是连接客户端和服务端间的通信,客户端的业务请求通过通信层发送到业务调用层[13]。

(3)业务调用层的主要作用是根据客户端的请求调用服务端的业务逻辑,业务逻辑处理完之后返回给 RMI 通信层,再由 RMI 通信层返回给客户端[13]。

(4)业务配置层的主要作用是配置通信接口提供的业务类型。当启动程序时,业务调用层会读取通信接口的业务配置信息[13]。

在图 1 中通过加入了 AOP 框架,这是一个间接增加额外层,它包含了一系列自适应的方面类,比如实现了非功能性横切关注点分布性的方面类,该方面类将在运行时自动整合功能性代码与分布性的非功能代码,这些方面类横切组件和整个系统。

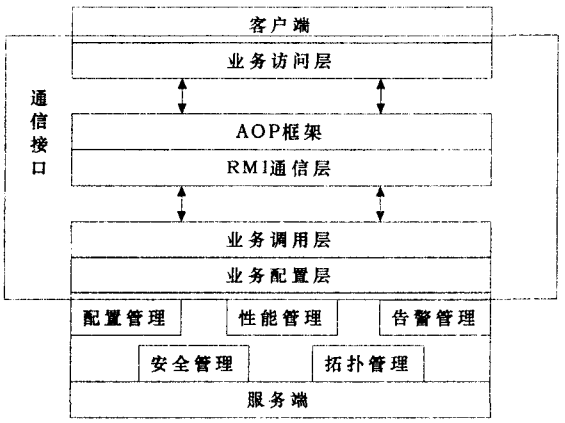

图 1 RMI 通信接口体系结构

3 应用面向方面的编程思想对中间件进行重构

下面通过在一个分布式环境中,应用程序允许客户端从远程主机服务端下载文件为例来说明如何应用面向方面的编程思想对中间件进行重构。

使用 RMI 开发一个分布式应用包括如下几个步骤:

(1)定义一个远端的接口。

(2)实现这个远端的接口。

(3)开发一个服务端。

(4)开发一个客户端。

(5)生成 Stubs 和 Skeletons,运行 RMI 注册器,服务端和客户端。

从以上的步骤中可以看到,当客户端每一次进行对服务器的操作时,一方面都得先考虑分布式应用的需要,另一方面由于任何远程方法调用实际上要进行许多低级网络操作,网络错误可能在调用过程中随时发生,因此还得对每一次远程的操作异常进行处理。应用面向方面的编程思想对中间件的分布性进行重构,通过把此类中间件特有的代码,如分布性代码完全封装到方面类代码中,这使得通过为任何一个中间件写专有的方面,然后用相同的构件代码来支持很多不同的中间件成为可能。

按照面向方面编程的编程思想,可以把功能代码和非功能代码分离开来,这样在客户端从服务端下载文件的例子中包含了纯功能的构件代码,而编写这些代码并没有耦合分布式中间件通信环境要求的代码,其代码如图 2 所示。

```
/**
 *客户端从远程服务端下载文件
 */
publc class Filelmpl
    //从远程主机下载文件的核心功能代码
    public byte[]downloadFile(String fileName){
        ……
    }
    //其它业务逻辑需求
    ……
}
```

图 2 不含分布性的纯功能代码

可以把客户端的远程分布式操作应用面向方面的编程思想构建成方面,这样客户端要对远程资源的操作只要考虑功能性代码。

图 3 中表明了一个应用面向方向编程思想客户端的远程访问中间件。

```
/* *
* 采用 AspectJ 风格的 AOP,使用 Annotation 注解来定义一个通知
* @Aspect 声明一个切面,@Component 注册受管 Bean
* @Around 声明一个通知
* /
@Aspect
//声明一个切面
@Component
//注册受管 Bean
public clsaa FileAspect{
    //"@Pointcut"注解声明了一个切入点,它匹配了 FileImpl 类中的所有方法
    @Pointcut("execution( * FileImpl * (..))")
    public void pointcut(){
    }
    //声明一个环绕通知,该通知利用 JointPoint 参数访问连接点
    @Around("pointcut()")
     public Object aroundAdvice (ProceedingJoinPoint point) throws
Throwable{
        //分布式环境的处理逻辑
        ......
        Object o = point. proceed(point. getArgs());//调用连接点方法
        return o;//把连接点的返回值返回给调用者
    }
    //异用通知,使用 throwing 获取连接点异常
     @ AfterThrowing (pointcut = "pointcut ( )", throwing = "throwing")
     public void afterThrowingAdviec (RemoteException throwing) throws
Throwable{
        System. out. println("RemoteException 通知" + throwing);
    }
}
```

图 3　客户端的方面类代码

在这个实例中,方面类 FileAspect 中定义了一个切入点,它匹配了 FileImpl 类中的所有方法,此处也是影响图 1 功能代码的地方。我们定义了二个通知,名叫 Around 的通知在切入点处织入了分布式通信与图 1 的功能代码,名叫 AfterThrowing 的通知在切入点处织入了远程操作异常与图 1 的功能代码。

因此,也可以应用面向方面的编程思想,重新构建服务器端的中间件。

4　结束语

实时系统中间件中,功能代码和非功能代码是紧密耦合的,这样大大影响了中间件体系结构的可移植性和可配置性等一系列问题。为了解决这些问题,利用了面向方面的编程思想对中间件进行了重构。面向方面是模块化横切关注点的一种方法,它的设计思想是采用一种松散耦合的方式来实现系统非功能的关注点,然后在系统运行时织入关注点。

文中主要是对基于 Java 的 RMI(Remote Method Invocation)分布式框架进行通信的实时系统中间件,应用面向方面的编程思想来重构分布性,中间件在编写分布式应用时,只要关注它的核心功能代码,之后通过方面编织器织入分布式通信的非功能代码。这使对中间件的开发一方面有效地控制软件的开发风险及开发周期,另一方面也提高软件系统的可靠性、可扩展性、可重用性和可维护性,并大大降低系统的复杂性。

参考文献:

[1] Zhang C, Hans - Arno J. Refactoring Middleware with AspetJ [J]. IEEE Transactions on Parallel and Distributed Systems, 2003,14(11):1 - 16.

[2] 董华山,孙济庆,吉久明. AOP 在数据访问中间件中的应用[J]. 计算机应用与软件,2005,22(1):53 - 54.

[3] 李　行,张立臣,陈　成. 面向方面的实时系统中间件[J]. 计算机技术与发展,2008,18(7):8 - 10.

[4] 刘瑞成,张立臣. 实时系统的面向方面模型[J]. 计算机工程与设计,2006,27(6):937 - 940.

[5] 刘敬勇,张立臣,钟　勇. 面向方面的中间件[J]. 计算机技术与发展,2008,18(8):68 - 71.

[6] Loughran N, Parlavantzas N, Pinto M. Survey of Aspect Oriented Middleware[C]//AOSD - Europe Project Deliverable. [s. l.]:[s. n.],2005.

[7] Zakaria A, Hosny H, Zeid A. A UML Extension for Modeling Aspect - Oriented Systems[C]//Proceedings of International workshop on Aspect - oriented modeling with UML. Germany:[s. n.],2002.

[8] 原慧琴,贾杏丹. 面向方面的数字图书馆的开发研究[J]. 电脑与电信,2008(9):77 - 79.

[9] 康　蕊,张立臣. 基于 AOP 的 QoS 中间件自适应机制研究[J]. 计算机科学,2008,35(8):287 - 289.

[10] 陈志刚,李　登,曾志文. 基于 Java RMI 技术的中间应用服务器动态负载均衡模型的实现[J]. 计算机工程,2001,27(7):48 - 50.

[11] 陈景燕,阳国贵. AOP 下的权限控制实现[J]. 计算机与信息技术,2006(6):48 - 50.

[12] 侯　丞,康建初. 基于 CORBA 的电信网络性能管理接口[J]. 计算机工程,2006(2):257 - 259.

[13] 杨　萍. 基于 AOP 和 RBAC 策略的访问控制的实现[J]. 计算机与数字工程,2008,36(9):195 - 197.