MOCHA中自部署代码重用的研究

谢振华,朱跃龙
(河海大学,江苏 南京 210089)

摘 要:由于企业内互联网的增多,新的应用功能要求被加入系统,为了适应这种变化,出现了一种新的基于元数据的可自扩展的数据库中间件MOCHA。然而这种系统不能使部署的代码重复使用,造成了资源的浪费。文中分析了MOCHA的基本功能和要自部署的代码的特性,以及这些代码重用的可行性;提出一种基于元数据的代码重用的解决方法,使得代码当需要重用时,能提供一种定位可重用组件的机制。另外,可以掌握所有可重用代码的现状和历史,利于维护。

关键词:元数据;查询处理协调器;数据访问提供程序;资源描述框架
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2006)07-0114-03

Research on Code Reuse in Auto-Deployment of MOCHA

XIE Zhen-hua, ZHU Yue-long
(Hohai University, Nanjing 210089, China)

Abstract: Have proposed a novel metadata-driven based on database middleware MOCHA which can deploy application-specific data types and query operators automatically. First present the kind of services, the kind of publishing resources. Then analyze the characteristics of the code and the possibility of the code reuse. Finally give a resolve to code reuse.

Key words: meta-data; QPC; DAP; RDF

0 引 言

随着3W的普及,因特网的不断壮大,企业内部互联网的数量也在增加,因此需要数据中间件来互联许许多多通过这些网络来部署的数据站。存储在这些站中的数据集是以复杂的数据类型为基础的。在许多特定领域中,一些新的特定应用操作和数据类型会逐渐地添加到系统中,这时在数据库中间件中的全局模式必须发生相应的改变来反映系统中的变化。由此一种新颖的基于元数据驱动的具有自部署特定应用功能的中间件系统MOCHA[1,2]诞生了。

然而在这个系统中,部署到远程的代码无法实现重用,需要时必须重新迁移,已迁移的代码只能遗弃,导致可再用资源的浪费,增加了网络传输的开销。文中首先概括描述了MOCHA中的各种服务、资源发布的种类,即其代码的自部署功能。然后,分析了自部署的代码的特性,以及这些代码重用的可行性。提出一种代码重用的解决方法。

1 MOCHA的体系结构

首先介绍一下MOCHA的重要组件和体系结构。MOCHA是建立在两个基本的原则基础上的,这两个原则能使它克服以前中间件方法的缺点。第一个原则是所有进行任意给定的查询所需要的特定应用的功能将由MOCHA自动地传送给所有有需求的站,这将通过移动包含需要功能的JAVA类实现。第二个原则是每个查询运算符将在引起数据移动最少的站计算出。

如图1所示:最顶部是客户端应用程序,它向用户提供图形界面,向系统提出查询请求,以及向用户展示结果。客户端是通过URL连接查询处理协调器(QPC)。QPC是一个服务程序,它提供基本的查询处理服务和部署所有的必须的特定应用功能代码。QPC的主要功能是:查询解析、元数据管理、查询优化、代码部署、查询执行、错误管理。

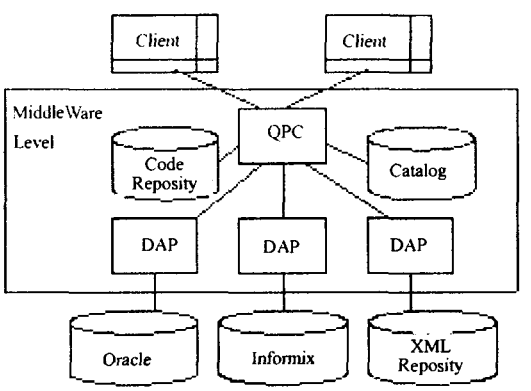

图1 MOCHA体系结构

收稿日期:2005-10-10
作者简介:谢振华(1981-),男,江苏苏州人,硕士研究生,研究方向为计算机软件与理论;朱跃龙,教授,研究方向为软件复用与软件构建技术、数据库技术。

为了访问存储在数据源里的大量的信息,QPC通过数据访问提供程序(DAP)来访问数据。每个数据源至少有一个DAP,而QPC也是通过URL来定位DAP的。DAP主要提供两种服务:

a.数据转换:DAP从数据源中获得数据项,把它们转换成QPC中使用的全局模式;

b.查询处理:DAP可以执行查询操作来进一步提取数据,它可以用来过滤冗余数据,因此DAP应该运行在数据源附近。

QPC传输所有的数据类型代码以及操作符到DAP。同样,所有DAP产生的结果也会返回到QPC进行进一步处理得出最终结果。

2 代码部署研究

2.1 资源发布

在MOCHA中,QPC就是通过从资源中获得的元数据中得到有用信息。这些资源包括表、查询操作符和数据类型。这些资源是建立在RDF的基础上的。每个资源用一个URI来标示。MOCHA利用资源的URI作为查找目录里元数据的查询键,原数据包含在RDF文档中。对于每一个资源。管理员通过目录管理工具可以把元数据入口以(URI,RDF File)的格式存入目录的表中。

2.2 代码部署

代码的自动部署在QPC接收到客户新的查询请求后就开始。QPC的第一个任务是生成查询进程相关数据类型和操作的列表。然后QPC访问目录中的元数据来将类型和操作映射到特定代码实现。每个代码库中的类被QPC的装载器回调后准备分发。在实际的查询执行前,QPC分配任务给运行在目标数据点上的每个DAP来执行。然后,QPC开始代码部署阶段,先将数据类型所需的类迁移到客户端和DAPS,再是查询操作的类[3]。一旦代码部署阶段完毕,QPC通知每一DAP激活其查询计划的块,通过反射技术,把迁移的类动态载入到JVM中,QPC和DAP就开始处理数据和产生结果,数据和结果将由QPC收集并且发送回客户端。

特别的是代码部署完全是自动的没有人工干预。在代码部署期间,在能使用收到的功能前,不需要重起MOCHA中的任何元素。相反,每个QPC和DAP以及客户应用都具有装载类和立即执行的必需逻辑。因此,MOCHA中的每个元素都能在运行期间以动态方式扩展。这样,管理员的任务大大简化了,他们只要处理一个或几个代码库。新的和更新的功能只要简单地加到库中,根据需要被部署。

3 代码重用的可行性

3.1 自部署代码的特性

3.1.1 用户自定义操作运算符的类

在MOCHA中,查询操作符分为两类:复杂方法和聚合。

a.复杂方法是指在查询中包括复杂谓词和投影。复杂方法在JAVA类中是通过静态方法来实现的。

b.聚合操作是通过JAVA类的实例来实现的。这种类必须实现聚合标准接口。

3.1.2 用户自定义数据类型的类

要自动部署到DAP上的类还有用户自定义的一些数据类新的JAVA类,一个类就影射了对象数据库中自定义的一种数据类型[4]。图2表现了MOCHA原型的类型系统的继承结构。黑矩形代表类型接口;白矩形代表JAVA类为特殊的类型。层次的根是MWObject接口,该接口定义类为一个MOCHA数据类型的实现,并定义了在网络中传递对象实例的方法。

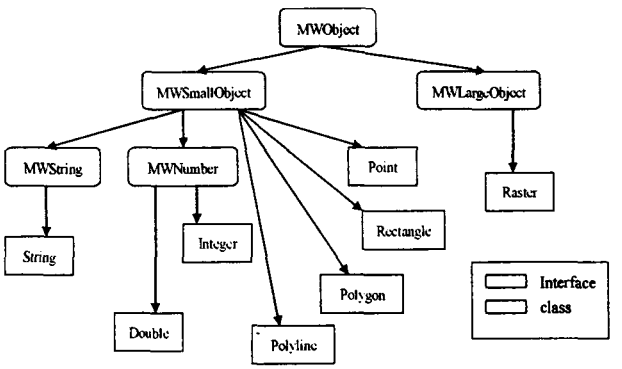

图2 数据类型

MWSmallObjet和MWLargeObject接口是MWObject的继承,将所有的类型分为两组:大对象和小对象。大对象用于诸如图像、音频和文本文档等大的数据类型;小对象用于诸如数字、字符串、点或者长方形等较小的数据类型。另外,字符和数字类型的接口起源于MWSmallObject。任何被添加来系统的新的类型必须在MWObject之下实现接口。

3.2 代码重用的可行性分析

这两种类型的JAVA类与一般的类有区别。更具这两种类型代码的特性,以下对它们分别做代码的重用可行性分析。以确定这两种类型的JAVA代码是不是有利于重用。

3.2.1 用户自定义操作运算符的类的重用可行性

对于操作符的类,把这些类存放在一个类库中。其中复杂方法会被其他方法调用,或者直接被执行引擎执行产生结果。要实现复杂方法的重用。继承机制不是一个理想的选择。因为继承的时候必须继承一个类的所有数据成员和所有的方法,否则就无法重用这个类的某个单独的方法。这些额外的不必要的负担使方法重用的代码变得复杂。派生类对其父类的依赖性也移入了额外的复杂性:对父类的改动会对子类造成影响;当修改任意一个类的时候,很难记得清哪个方法被覆盖,哪个没有;而且被覆盖的方法是否会调用父类中相应的方法并不非常清晰地显现。

任何执行单一概念任务的方法应该能够成为代码用

的首选而独立存在。为了达到这个目标,必须回到过程化的编程模式,将代码从类实例的方法中移出,形成具有全局可见性的过程。为了提高这种过程的可重用性,过程代码应该像静态的通用方法一样编写。这种对外部依赖的简化降低了过程使用的复杂性,也增加了在其他地方使用此过程的可能性。因而复杂方法在 JAVA 类中通过静态方法来实现是有利于代码的重用的。

聚合方法是通过一个类或者一组类实现的。因此,一个类或者一组类确定了一个方法,并且该类或该组类与其它类之间没有继承和被继承关系。当 DAP 要加载某个特定的方法时,只需要加载该方法所在的一组类。这组类的改变和更新都不会影响类库中其他的类,由此可知该类库具有低耦合性、细粒度性,有利于代码的重用。

3.2.2 *用户自定义数据类型的类的重用可行性*

在代码中会出现大量的参数,这些参数的类型有些是用户自定义的,如上所说,这些类型实现了统一的几个接口。因此,可以将参数类型改为接口类型,在面向对象编程中,代码重用的真正基础在于通过接口参数类型利用多态性,而不是通过类继承。同时,还要选择低耦合的参数接口类型。参数对象所要实现的接口越简单,其他特定类实现此接口的机会就越大,由此,其对象可以作为参数使用的类也就越多。由图 2 可知,所有自定义的新类型都在 MWObject 下实现接口,因此,用户自定义数据类型的类也是适合重用的。

4 代码重用的解决方案

对于这两种类型的代码,提出了一种解决的方法就是让 QPC 和 DAP 交换已经传递到的最新更改日期的信息。DAP 告诉 QPC 日期不匹配的实例,QPC 只要传递这些代码就可以了。而这些原始信息则是在 QPC 向 DAP 迁移代码时记录下的,这些信息是描述了数据库的名称以及相应 DAP 上拥有最新的类的信息。这些信息也存储在资源中用 RDF 文档来描述。以下给出试验步骤:

(1)QPC 把日期的信息存入 RDF 文档。

RDF (Resource Description Framework,资源描述框架)是一个用于表达关于万维网上的资源的信息的语言[5]。它专门用于表达关于 Web 资源的元数据,比如 Web 页面的标题、作者、修改时间、Web 文档的版权和许可信息,某个被共享资源的可用计划表等。

该 RDF 保存了迁移代码的类名、它的创建日期和它的修改日期以及其他一些有用的元数据。以便于将来 QPC 重用代码进行判断。

(2)采用 RDF 解析器(比如 jena)解析文档,进行日期匹配。

通过解析 RDF 文档,同原来已传递到 DAP 上的代码的副本进行比对,这里会出现以下 3 种情况:

a. 如果 RDF 中记录的代码的时间与副本中相同名称代码的修改时间不同,则说明这则代码已经被修改过,必须重新迁移这部分代码;

b. 如果 RDF 中记录的代码的时间与副本中相同名称代码的修改时间相同,则说明这些代码没有改动过,可以直接重用 DAP 上的原有代码;

c. 如果 RDF 中记录的代码在副本中找不到相同名称的代码,则说明这则代码还未被迁移到 DAP 上,需要迁移这部分代码。

(3)迁移代码。

这部分通过 JAVA 的输入输出流来实现。在 JAVA 中所有的文件都是字节形式的。在 QPC 端通过 FileInputStream 获得一个输入流,把要迁移的 JAVA 代码读入到流中,通过远程方法调用,把该流的对象传递到 DAP 上,再在 DAP 上通过 FileOutputStream 把流中的数据读出来并且保存在相同名称相同类型的文件中,这样就实现了代码的迁移。

(4)动态加载迁移的代码。

通过 JAVA 自省能力从元数据中获得相应 JAVA 的类及其对象,还有该类中的有用方法[1]。再不重新启动服务器和重新启动应用程序的情况下动态地把这些类和方法加载到 JAVA 虚拟机(JVM)中。

JVM 中有一个缺省的 ClassLoader,它只知道如何从本地文件系统装入类文件。不过这只适合于常规情况,即已全部编译完 JAVA 程序,并且计算机处于等待状态。但可以使用定制的 ClassLoader 完成动态地创建符合用户特定需要的定制化构建类。为此将创建 CompilingClassLoader 的 ClassLoader。它为人们编译 JAVA 代码,而无需要人们干涉这个过程。

通过以上方法就能实现从判断代码是否能重用,若不能被重用的则迁移到 DAP 上,以及这些新迁移的代码和可以被重用的代码最后动态地加载到系统中去的全过程。

5 结束语

由于在现存的中间件解决方案中,用户为数据类型和查询操作定义的功能完全是手工发布的。在拥有上百个数据源的环境下,系统管理员要为每个站点安装和维护必须的软件库,从而造成了很高的开销和复杂性。在 MOCHA 中,用户定义的功能被系统自动发布到那些不提供次功能的站点中。这是通过将那些实现所需功能的 JAVA 类装载到远距离站点来完成的。而 DAP 能够重复使用迁移的代码的研究。可以使传输的代码有效地进行重用,增加了资源的使用率,减少了网络传输的开销。将来的工作包括对代码重用及迁移的安全性研究,以及该系统在实际领域中应用的研究。

参考文献:

[1] Venners B. 深入 Java 虚拟机[M]. 北京:机械工业出版社,2003.

(下转第 188 页)

表 1 三组图像的配准参数

切片图像基准	配准参数				
	平移因子 $\triangle x$	平移因子 $\triangle y$	旋转因子	比例因子 k	转角 θ
MRI 图像(1)变换	6	-1	0.063695	1.0338	1.8214
MRI 图像(2)变换	2	9	0.046563	0.99831	1.2208
MRI 图像(3)变换	-8	2	0.048364	0.94066	1.3484

第二组:如图 3 所示(其中,左为未配准前叠加图,右为配准后叠加图)。

第三组:如图 4 所示(其中,左为未配准前叠加图,右为配准后叠加图)。

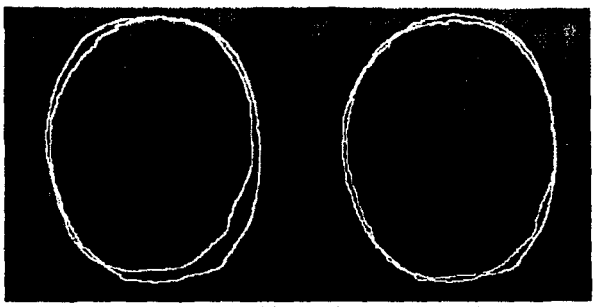

图2 第一组切片图像配准前后叠加图对比

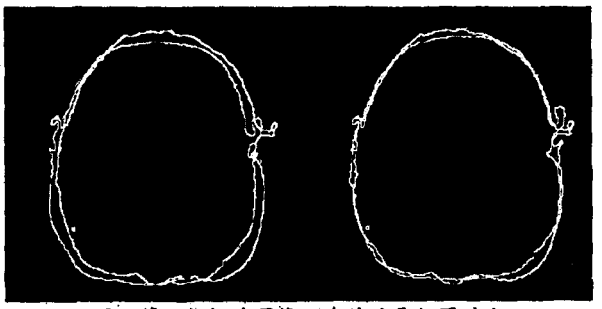

图3 第二组切片图像配准前后叠加图对比

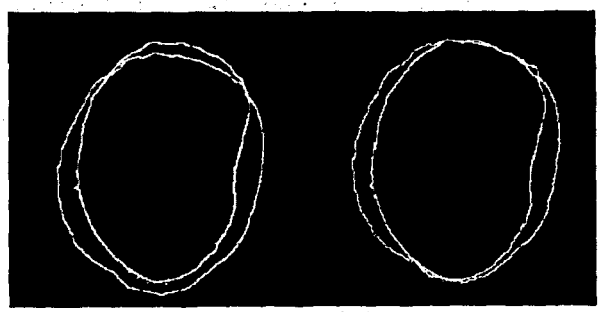

图4 第三组切片图像配准前后叠加图对比

4 结 论

对数字人脑部多模图像进行基于轮廓的几何特征的配准能够实现,且此算法配准速度快、效果明显。对现有数字人头部图像的配准结果基本令人满意,其结果可以作为后期图像融合工作的依据。

但是,在对实验的处理过程中发现此配准方法中,仍存在需要改进的方面:

1)由于数字人不同图像的成像原理和图像采集过程不同,CT/MRI 图像与数字人图像的切面明显存在着倾角,使得在配准时如何定位同一层面不同模态图像和如何修正图像平面间的夹角成为一个必须考虑的问题。

2)数字人切片图像的采集和 MRI 断层图像的采集在物理过程上不属于同一时间段和物理过程。其结果就是数字人切片图像的头部轮廓与同层面 MRI 断层图像的头部轮廓存在型变。如何对该形变进行修正也是笔者后继研究的方向。

3)由于求轮廓几何特征进行配准的方法强烈依赖于两幅待配准图像感兴趣区域的轮廓特征,因此图像轮廓的不均匀或轮廓的不连续将直接影像几何特征的求取。也就是说,使用该方法对图像进行配准时,求取不同轮廓的过程需要尽量保证两幅图像轮廓的几何特征(如有效像素点)的相近。

参考文献:

[1] 钟世镇,李 华,罗述谦,等. 中国数字化虚拟人研究[A]. 香山科学会议第 174 次学术讨论会文集:"中国数字化虚拟人体的科技问题"[C]. 北京:[出版者不详],2001.4-12.

[2] 钟世镇. 数字化虚拟人体在医学应用上的前景[J]. 中华实用医学,2002,4(9):1-3.

[3] 钟世镇,牛憨笨,李 华,等. 中国数字化虚拟人体研究的发展与应用[A]. 香山科学会议第 208 次学术讨论会文集:"中国数字化虚拟人体研究的发展与应用"[C]. 北京:[出版者不详],2003.5-11.

[4] Lee Y S, Chung M S, Hwang W S, et al. Visible Korean Human: Another Trial for Making Serially-Sectioned Images [A]. Proceedings of The Fourth China-Japan-Korea Joint Symposium on Medical Informatics [C]. Beijing: [s. n.], 2002.168-174.

[5] 罗述谦. 医学图像配准技术[J]. 国外医学生物医学工程分册,1999,22(1):1-7.

[6] 吴 锋,钱宗才,杭洽时,等. 基于轮廓的力矩主轴法在医学图像配准中的应用[J]. 第四军医大学学报,2001,22(6):567-569.

(上接第 116 页)

[2] Rodriguez-Martinez M, Roussopoulos N. Automatic Deployment of Application-Specific Metadata and Code in MOCHA [R]. UMIACS-TR 2000-05, CS-TR 4105. MD, USA: University of Maryland, 2000.

[3] Rodriguez-Martinez M, Roussopoulos N. MOCHA: A Self-Extensible Database Middleware System for Distributed Data Sources[R]. UMIACS-TR 98-67, CS-TR3955. MD, USA: University of Maryland, 1998.

[4] 莫里斯奥·利诺埃. Qracle9i SQLJ 程序设计[M]. 北京:机械工业出版社,2002.

[5] Lassila O, Nokia Research Center, Swick R R. World Wide Web Consortium[EB/OL]. http://www.w3.org/Archives/Public/www-rdf-comments, 1999-02-22.