

事务监控器的 Web 服务支持^{*}

丁晓宁 张 昕 金蓓弘

(中国科学院软件研究所软件工程技术中心 北京100080)

摘 要 事务监控器是开发、部署、监控和管理分布式事务性应用的基础平台。随着 Internet 和企业应用集成的发展,许多应用要求能通过广域网访问部署于事务监控器上的服务,同时能与其它分布式系统互操作,但传统的事务监控器作为一种封闭的结构,对此支持不足。Web 服务是一种新的分布式组件标准,可方便地部署于广域网上并灵活地进行企业应用集成,本文提出了事务监控器支持 Web 服务的一个可移植的设计框架 TP-WS,可以在符合 X/Open DTP 模型的事务监控器中支持 Web 服务。本文设计现已应用到事务监控器 ISTX2.0 中。

关键词 事务监控器, Web 服务

Web Service Support in TP Monitor

DING Xiao-Ning ZHANG Xin JIN Bei-Hong

(Technology Center of Software Engineering, Institute of Software, Chinese Academy of Sciences, Beijing 100080)

Abstract TP Monitor is the platform for developing, deploying, monitoring and managing three-tier applications. As the development of Internet and enterprise application integration, demands of accessing services deployed on TP Monitor have been raised, and TP Monitors also need to interoperate with other distributed systems. However, traditional TP Monitor lacks of support of these needs. Web Service is a kind of new distributed component standard, and it can be deployed on Internet or integrated with enterprise applications easily. The above problems can be solved with the support of Web Service by TP Monitor. In this paper, we present a portable framework TP-WS, which can be used in traditional TP Monitor to get full support of Web Service. Now the design has been successfully implemented in a TP Monitor, ISTX2.0.

Keywords TP monitor, Web service

1 引言

事务监控器(Transaction Processing Monitor)是开发、部署、监控和管理基于三层结构的事务性应用的基础平台^[1,2]。在事务监控器中可部署和运行业务逻辑(Business Logic)封装成的服务,事务监控器负责提供事务处理、消息通信、负载均衡、失败恢复等功能,从而大大减轻了应用开发人员的负担。事务监控器已在银行、证券、航空等行业得到广泛应用。这些应用一般构成一个封闭的体系,即客户和服务器或者在同一个局域网中,或者通过专用网络连接。

随着 Internet 与企业应用需求的快速发展,出现了如下几个新问题:

第一,Internet 的迅速发展和企业业务开放性的增强,有大量客户需要通过 Internet 访问事务监控器,如银行系统会发布“个人银行助理”等应用,使客户在本地即可查询历史交易,打印帐单等。另外,企业应用集成也要求能跨越广域网访问事务监控器。然而,部署在传统事务监控器上的应用不适合在广域网上访问。因为 Internet 上分布着大量的防火墙,它们通常只允许一些固定端口的数据包通过,而事务监控器客户端和服务器使用的通信端口是运行时系统动态选择的特殊端

口号,通信信息跨越防火墙时会被拦截。现有的解决方法是通过 Web 服务器再访问事务监控器,然而,Web 是无状态的,并不适合交易处理,更重要的是没有事务来保证交易的完整性和一致性。同时,Web 的页面形式只适合浏览,不适合企业应用集成等需要程序自动处理的场合。所以需要提供一种方法,能使客户越过 Internet 上的大量防火墙访问事务监控器。

第二,客户程序为了访问事务监控器,必须安装有该事务监控器的客户端支持,通常是一些针对特定平台和语言的函数库与命令行工具。这样就限制了客户端平台和语言的选择范围,例如有的事务监控器可能只提供了 Windows 平台的动态链接库,这样就无法使用 Unix 下的语言(如 Perl 等)来编写该事务监控器上的客户程序。传统局域网环境下,企业内部可能使用的客户平台和开发语言数目有限,这个问题并不突出,然而,一旦应用扩展到了 Internet 上,客户平台和语言就会不胜枚举。限于开发力量,一个事务监控器系统想要支持所有的平台和开发语言是不现实的。因此有必要提供事务监控器的一个通用接口,使得 Internet 上的绝大多数平台和语言都可以调用其服务。

第三,随着其它分布式体系的出现和成熟,一个企业通常有大量应用是基于 CORBA, COM/DCOM, J2EE 等规范开发

^{*} 本文研究得到国家863高科技发展计划资助项目(编号2001AA113010, 2003AA115440);国家重点基础研究发展规划973资助项目(编号2002CB312005)的资助。丁晓宁 博士生,主要研究领域为网络分布计算和软件工程;张 昕 博士生,主要研究领域为网络分布计算和软件工程;金蓓弘 博士,副研究员,主要研究领域为网络分布计算和软件工程,数据库实现技术。

的。为了保护用户投资,基于事务监控器的系统应当能调用这些已经开发好的业务逻辑,而且自己的服务也应当能被上述分布式系统调用。同时,企业应用集成等也要求各个分布式系统能实现互操作。而现有的事务监控器系统在此方面不够成熟,仅有的一些解决方法也是针对特定问题和特定产品。

Web 服务的出现,为上述问题提供了通用和可移植的解决方案。Web 服务是一种使用 Internet 标准协议的、松散耦合的分布式组件,Web 服务使用标准的端口号进行传输,可以轻松地跨越防火墙^[3];同时不必再针对每种可能的客户平台和开发语言提供支持环境,只需能够连接 TCP/IP 和解析 XML 即可,而这两个条件几乎所有平台和语言都能满足,最后, CORBA, COM/DCOM, J2EE 等分布平台都提供了与 Web 服务的互操作。

本文描述了事务监控器的 Web 服务支持框架 TP-WS。第2节给出了 TP-WS 框架的设计;第3节介绍了设计中的几个关键问题;第4节是 TP-WS 在 ISTX2.0 上的应用与性能分析;第5节是相关工作;最后给出了全文的小结。

2 TP-WS 设计框架

TP-WS 是一个事务监控器的 Web 服务支持框架。它适用于任何符合 X/Open DTP^[4,5]模型的事务监控器。X/Open DTP 是 X/Open 组织制定的事务监控器的规范,是事务监控器广泛遵循的一个工业标准。TP-WS 提供了事务监控器 XATMI 接口的包装,如果需要将 TP-WS 移植到一个符合 X/Open DTP 模型的事务监控器上,只需将 TP-WS 与该事务监控器提供的 XATMI 运行库及 Typed Buffer^[4]目标文件链接即可。

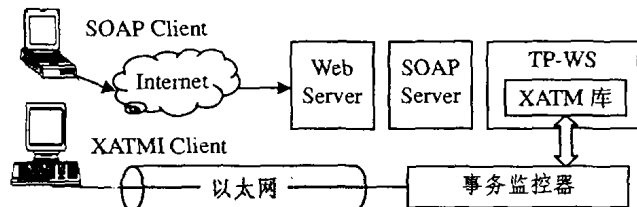


图1 TP-WS 和事务监控器关系

图1给出了 TP-WS 和事务监控器的逻辑关系,TP-WS 需要利用 Web Server(我们选用了 Apache Tomcat)和 SOAP Server(我们选用了 Apache Axis^[6]),监听并解析到来的 SOAP(Simple Object Access Protocol)消息。

TP-WS 的内部结构分为三层:运行支持层,连接管理层和服务接口层,参见图2。

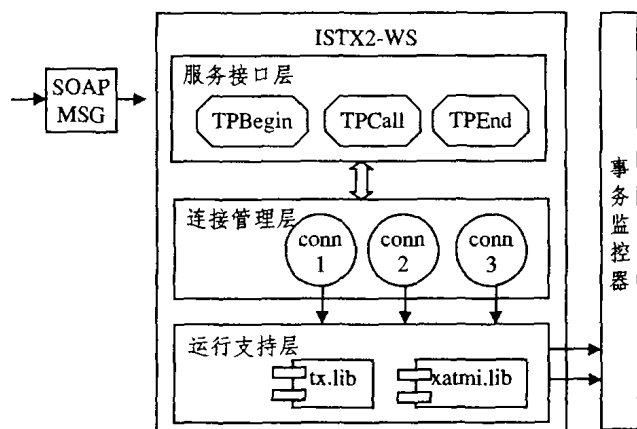


图2 TP-WS 的分层结构

运行支持层主体是事务监控器的客户端支持库,由事务监控器提供,完成的工作包括和事务监控器连接,创建和提交事务,调用服务等。该支持库符合 X/Open XATMI^[4]和 X/Open TX^[5]规范。运行支持层包含的另外一个模块就是发布服务所需的 Typed buffer 目标文件,该目标文件负责转换调用参数的网络字节顺序,可以由事务监控器自动产生。

连接管理层对运行支持层的功能进行了包装,抽象出 Connection 对象,每个 Connection 对象代表一个活跃的 SOAP 客户端。连接缓冲池管理,超时检测等主要功能均在此层实现。它是一个独立的应用程序,类似一个守护进程,通过 Socket 和服务接口层各个 Web 服务通信,提供给上层如下接口:创建一个 Connection 对象,设置属性,调用一个服务,关闭对话等。

连接管理层实现了 TP-WS 的主要功能,该层主要包括 Manager、ConnPool、Connection、IOChannel、Config 等类,图3给出了该层主要的类以及它们之间的关系。

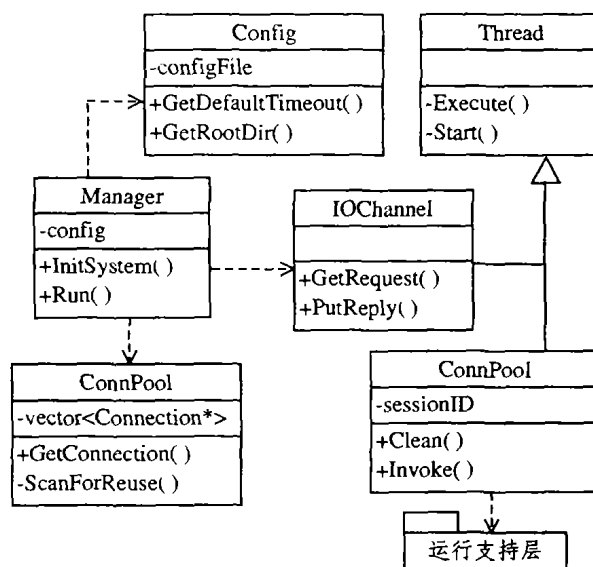


图3 连接管理层主要类图示

Connection 类封装了一个与事务监控器的连接,每个连接作为一个单独的线程处理。由于每个连接上,向事务监控器提交的请求都会有等待网络传输和服务处理时间,因此多线程可以避免空等,大大提高系统的吞吐量。

连接池(ConnPool)类管理所有当前活动的 Connection 对象,负责其创建、调度和销毁。IOChannel 类封装了消息的收发功能,它也是一个独立的线程,循环地从 socket 取请求消息以及返回应答消息。管理器(Manager)类维护消息主循环,它从 IOChannel 类获取请求消息,再将它派发给对应的 Connection 对象,最后将返回消息发送到 IOChannel 中。配置(Config)类封装了系统中所有的配置选项。

由于客户可能很多,而且与事务监控器的连接初始化消耗较大,为提高效率,TP-WS 使用连接缓冲机制。当一个 SOAP 客户调用 TPTerm 结束会话时,所对应的 Connection 对象并不释放,连接也不断开,而是执行一个 Clean 操作后标记为可用,放在连接池中等待下一次使用,连接池的大小限制可以在配置文件中指定。同时,当无用连接回收(Garbage Collection)线程发现有的 Connection 对象最近一次访问时间与当前时间之差已经达到设置的超时值,执行一个 Clean 操作,放入池中等待重用。如果在 Connection 释放后,再次收到对应的 SOAP 客户端请求,连接池查找不到对应的 Session-

ID,会返回错误,指出连接超时,资源已释放。

服务接口层是部署在 SOAP Server 上的一系列服务的集合,包括 TPInit,TPBegin,TPCall,TPCommit 和 TPTerm 等。按照 Web 服务的发现机制,这些服务的描述将以 WSDL (Web Service Description Language)文档的方式公布到 UDDI(Universal Description, Discovery and Integration),代表着整个事务监控器公布到 Web 服务上的接口。

在 TP-WS 中,每一个 Web 服务的客户端,在调用任何服务前,必须调用 TPInit 服务进行认证。服务器认证并创建连接后返回一个标志符 SessionID,在随后的所有调用中必须包含此标志符,SOAP 服务器根据此标志符维护客户端的连接上下文。

分层设计不仅使得结构更清晰,而且可以方便地增加和删除新的服务接口。同时,移植到其它符合 X/Open 规范的事务监控器平台,只需做很少修改。理想情况下只需简单地替换掉运行支持层,重新链接即可。

3 关键问题与解决

要在传统的事务监控器上实现对 Web 服务的支持,可能会出现以下一些问题,我们一一列举并给出 TP-WS 中对这些问题的解决方案。

1. 服务绑定方式。一个事务监控器要支持 Web 服务,可以将监控器中的每一个服务都包装成一个独立的 Web 服务,但是这种方案要求维护事务监控器、Web 服务支持模块、WSDL 三者间的一致性,而且和原事务监控器的内部结构密切相关,失去了可移植性。我们在 TP-WS 中提供一个独立固定的服务 TPCall,由 SOAP 客户在运行期间动态绑定所需的服务,既避免了上述弱点,又可以很自然地利用 XATMI 接口中的 TPCall()函数来实现。

2. 数据类型映射。X/Open XATMI 规范中,调用每一个服务,都需要传递一个参数结构类型(Typed Buffer),该结构必须预先定义好并调用事务监控器的工具编译,以产生对应的字节转换程序。SOAP 请求的服务不同,TP-WS 所需要使用的 Typed Buffer 也不同,在 TP-WS 中,有一个 Typed Buffer Manager,事务监控器的管理员配置好各个 Service 所对应的 Typed Buffer 类型,写入配置文件,并将每种 Typed Buffer 对应的目标库链接进来。运行时系统收到一个 TPCall 服务请求后,会动态查询 Typed Buffer Manager,获取该 Service 对应的 Typed Buffer。

3. 会话状态维护。传统事务监控器和客户的连接大多采用面向连接的协议如 TCP。这样,事务监控器可以在内部为每条连接绑定一个上下文。与此相反,SOAP 协议是无状态的,每一次 Web 服务的调用都是一次完整的会话,因此,必须维护调用上下文和多次调用间的联系。TP-WS 采用的方法是,在客户调用任何 Web 服务之前,先调用一个初始化服务,在此服务中进行用户名密码的验证,以及一些其他会话参数的设置(如会话超时值)。服务器进行认证并创建连接后,返回一个 SessionID。在随后的所有服务调用中必须包含 SessionID,TP-WS 根据该标志符维护该客户的上下文。

4. 安全问题。在事务监控器上开发的大多是一些安全敏感的关键应用,如银行、证券、航空等。传统的应用环境中,客户和事务监控器或者在同一个局域网中,或者通过专网连接,即使在这样的情况下,通信数据也需要经过加密。支持 Web 服务后,客户与事务监控器间的连接将跨越 Internet,并且

SOAP 的底层传输协议如 HTTP 或 SMTP 都是明文传送,显然这是无法接受的,无法抵挡简单如 sniffer 之类的攻击。TP-WS 利用了较成熟的基于 HTTP 的安全方案来解决,Apache Tomcat 可以配置为使用 SSL,在传输层保证了 Web 服务的安全。

5. 资源回收。传统事务监控器和客户的交互基于有连接的协议,这样一旦客户发生崩溃,服务器能够立刻得到通知,从而进行异常处理,回收资源。而 SOAP 调用的无状态性将引发资源回收问题,如客户在调用一个服务后突然崩溃,服务器是无法立即获知的,已经为该客户分配的资源将一直得不到释放。TP-WS 采用超时检查机制,每个 Connection 对象记录最后一次访问时间,由一个单独的线程间隔指定时间检查,如果已经超过初始化服务中设定的超时值,即释放它所占用的资源,SOAP 端再次发送请求时将收到“连接超时”的回复。

6. 扩展通信模式的支持。成熟的事务监控器支持与客户的多种通信模式,包括同步调用,异步调用,事件通知等。而 Web 服务的风格是典型的请求/应答格式同步调用,所以,一个支持 Web 服务的事务监控器如何支持扩展的通信模式,是一个必须考虑的问题。TP-WS 支持异步呼叫方式 TPACall。在服务接口层收到该请求后,将请求信息传给连接管理层,返回一个异步调用句柄,不再等待请求处理结束。SOAP 客户可随后在任意时刻用 TPGetReply 服务获取此次调用的结果。

7. 效率优化。Web 服务的优势在于跨平台和应用集成,但是由于 SOAP 消息以 XML 的文本方式编码,效率不如基于二进制的协议如 IIOP 高。同时,发送和接收前后的 XML 编码解码也会降低效率。由于一台 SOAP 服务器可能承载很多的客户,事务监控器的 Web 服务支持必须考虑效率问题并进行优化。TP-WS 使用了多线程和连接缓冲机制,由于与事务监控器进行连接初始化很耗费时间,因此连接缓冲机制大大提高了系统效率,同时,每个连接一个线程可以避免事务监控器处理请求时该连接的空等。

4 在 ISTX 2.0 上的应用与性能分析

ISTX 2.0 是中科院软件所开发的事务监控器,遵循 X/Open DTP 模型。在客户端主要提供了 xatmi.lib 和 tx.lib 两个 C 语言版本的运行支持库。

为了将 TP-WS 部署到 ISTX 2.0 上,首先由系统管理员配置需要导出到 Web 服务上的 Service 所对应的 Typed Buffer 类型,由事务监控器自动产生每个类型的目标文件,最后将它们和运行支持库,TP-WS 第二层目标文件链接,获得 TP-WS 的可执行文件。

安装 Apache Tomcat 和 Apache Axis 后,将 TP-WS 第一层的文件部署上去,SOAP 服务器将自动产生所有的 WSDL 文件。

图4是 TP-WS 应用在 ISTX 2.0 上,生成的 WSDL 文件片断:

```
- <wsdl:message name="TpCallResponse">
-   <wsdl:part name="TpCallReturn" type="xsd:string" />
- </wsdl:message>
- <wsdl:message name="TpCallRequest">
-   <wsdl:part name="sessionId" type="xsd:int" />
-   <wsdl:part name="serviceName" type="xsd:string" />
-   <wsdl:part name="inVal" type="xsd:string" />
- </wsdl:message>
```

图4 产生的 WSDL 文件片断

实际使用中发现,由于 SOAP 的文本传输协议,XML 解

析等特点,效率与原始的事务监控器 XATMI 库程序有较大的差距,必须针对特点做相应改进。表1是在 ISTX2上的 ToUpper 服务两种调用方式的延迟时间对比,ToUpper 服务用于将参数字符串改为大写。SOAP 客户端采用 Delphi 编写,广域网环境是通过局域网模拟得到,每次调用加上200~500毫秒的随机延迟,为减少误差,取循环运行100次的总时间。

表1 两种调用方式延迟时间对比

延迟时间(单位:秒)	SOAP	XATMI
局域网	5.4	0.2
广域网	39.7	34.8

SOAP 调用方式下,整个延迟时间 $t = \text{SOAP 处理时间} + \text{通信延迟} + \text{服务处理时间}$,比例系数 $p = (\text{SOAP 处理时间}) / (\text{SOAP 处理时间} + \text{通信} + \text{服务处理})$ 代表了 SOAP 处理在一次调用延迟中占用的比例。

具体到本例,局域网环境下,由于实验中采用的 ToUpper 是种计算量极小的服务,通信延迟也极小,经测量该比例系数约为0.91,绝大部分时间消耗在 SOAP 解析上,导致延迟时间远远大于 XATMI 方式。广域网情况下,由于通信代价的增大,该比例系数急剧变小,在本例中约为0.12,两种方式的性能差别就很小。比例系数越小,对 Web 服务调用方式越有利。

我们无法改变 SOAP 处理时间和通信延迟,但可以通过增大服务的粒度来降低比例系数。而事务监控器作为传统的交易处理环境,存在大量细粒度、计算量小的服务。所以在引入 Web 服务后,除了 TP-WS 本身的优化外,事务监控器必须根据 SOAP 协议的特点做相应的优化,如谨慎选择公布服务的类型,合并服务以增大粒度等,才能达到较满意的效果。

5 相关工作

目前市场上应用较广泛的事务监控器系统有 BEA 公司的 Tuxedo,IBM 公司的 CICS,Encina 等。目前它们对 Web 服务的支持均有不足之处。

Tuxedo 8.0 以前不支持 Web 服务^[9],在2003年发布的最新版本8.1中宣称支持 Web 服务,但是其支持是通过外接一个 J2EE 服务器(WebLogic)来实现的,从 Web 服务接口到真正的 Tuxedo 服务调用要经过 WebLogic 和 WebLogic Tuxedo Connector(WebLogic 到 Tuxedo 的连接程序)^[7],这样一方面非常低效,另一方面,如果需要使用 Web 服务,必须再另外使用 WebLogic 服务器,很大程度上增加了用户的成本负担。

IBM 提出一套解决方案^[8],即“SOAP for CICS”来使得部署在 CICS 上的应用可以包装为 Web 服务。SOAP for CI-

CS 的传输层可以使用 HTTP 协议或 IBM 消息队列 MQ Series。截至2003年4月份,该方案仍处于调研的阶段。而且目前只支持 OS/390,z/OS 这两个操作系统平台上的 CICS Server 版本,不够完备。

Encina 由 Transarc 开发,后被 IBM 收购,其开发工作早已停止,不支持 Web 服务。

目前这些传统的事务监控器的解决方案都是与对应厂商的产品具体相关的,不具备通用性和可复用性。

小结 事务监控器是开发部署运行分布式应用的重要平台,在银行、证券、航空等关键行业得到广泛应用。随着新的应用需求的发展,传统事务监控器暴露出很多问题,如不能部署基于广域网的应用,对 Internet 环境下的多平台和多语言支持不够,不能与其他分布式系统互操作等。Web 服务是近年来提出的一种分布式应用组件标准,如果事务监控器支持 Web 服务则可以解决上述问题。然而迄今为止,主流的事务监控器在 Web 服务的支持方面均显不足。我们提出了一个 Web 服务支持框架 TP-WS,它可以移植到任何符合 X/Open 标准的事务监控器上,目前该设计已经成功应用于我们自主开发的事务监控器 ISTX2.0 上。

TP-WS 应用于事务监控器后,事务监控器就可以融入面向服务的体系结构(Service-Oriented Architecture)中,作为服务提供者和服务请求者出现,可以解决引言中提到的三个问题以及其它一些新问题。但事务监控器与 J2EE,CORBA 等其它分布式系统的互操作目前还只是一种浅层次的互操作,这其中的事务上下文和安全上下文的跨系统传播等,还有待进一步的研究。

参考文献

- 1 Cray J, Reuter A. Transaction Processing: Concept and Techniques. Morgan Kaufmann Publishers, Inc. 1993
- 2 Bernstein P A, Newcomer E. Principles of Transaction Processing. Morgan Kaufmann Publishers, Inc. 1997
- 3 Apshankar K, Ayala D. Professional Open Source Web Services. Wrox Press Ltd. 2002
- 4 X/Open Company Ltd. Distributed Transaction Processing: The XATM Specification, 1991
- 5 X/Open Company Ltd. Distributed Transaction Processing: The TX Specification, 1991
- 6 Apache Group. <http://ws.apache.org/axis/index.html>
- 7 BEA. <http://edocs.bea.com/tuxedo/tux81/overview/webaces.htm>
- 8 IBM. <http://www-3.ibm.com/software/http/cics/soap/>
- 9 Andrade J M, et al. The Tuxedo System: Software for Constructing and Managing Distributed Business Applications. Addison-Wesley Publishing Company, 1996
- 5 Cockayne W R, Zyda M. Mobile Agents. Manning Publications Co., <http://flower.ce.cnu.ac.kr/~dkkang/mobile-agent-books/HTML/Front.htm>, 1998
- 6 Bakre M, Badrinath B. I-TCP: Indirect TCP for mobile hosts. In: Proc. 15th Intl. Conf. on Distributed Computing Systems, Vancouver, Canada, 1995
- 7 Brown K, Singh S. M-TCP: TCP for mobile cellular networks. ACM Computer Communications Review, 1998, 27(5)

(上接第143页)

- 2 Breugst M, Magedanz T. On the Usage of Standard Mobile Agent Platforms in Telecommunication Environments. available via www.fokus.gmd.de/research/cc/ecco/climate/intro-climate-cluster.html
- 3 ISO/IEC IS 10746-1 | ITU-T X.901, ODP-RM Part1: Overview, 1995
- 4 OMG. CORBA 3. Specification. <http://www.omg.org/>.