Web软件系统事务处理模型设计

田 冲,李兴国
(合肥工业大学 信息管理与信息系统研究所,安徽 合肥 230009)

摘 要:文中致力于把Web软件系统的结构模块化,通过建立新的事务处理模型以灵活的实现各功能模块间的组合,并引入服务进程以控制事务的执行流程。采用与面向服务架构整合各服务相反的思路,把系统的事务处理模型划分为服务注册中心、服务提供者、服务请求者、虚拟处理中心和用户交互中心五个角色。设计了各角色的工作方式并给出事务处理模型实现的关键问题及其解决方案。建立了处理Web软件系统灵活服务组合的事务处理模型,通过五个角色间的协调,实现了系统结构的模块化。应用实践表明该事务处理模型为设计灵活的Web软件系统提供了方案。

关键词:面向服务架构;进程;模块化;事务处理模型
中图分类号:TP39 **文献标识码**:A **文章编号**:1673-629X(2009)10-0062-04

Design of Web Software System Transaction Processing Model

TIAN Chong, LI Xing-guo
(Institute of Info. Management and Info. System, Hefei University of Technology, Hefei 230009, China)

Abstract: Attempts to make Web based software system modularized. Builds up a new transaction processing model to make different function parts work together effectively. Apply service process to control transaction status. Compares with the SOA integrating different service parts, classifies software system into five roles: service register center, service requester, virtual central processing unit, user interface center and service unit. Regulations about each role are designed and key technologies about the actualization of the transaction procession model are also analyzed in this paper. These five parts correspond together to get the structure of the system modularized. A transaction processing model is built up which can deal with flexible service combination. Application shows that it provides a method for the designing of flexible Web software system.

Key words: SOA; process; modularization; transaction processing model

0 引 言

随着互联网的发展,基于B/S架构的软件系统得到了广泛的应用。但随着企业业务的增加和工作流程的转变,系统的结构面临着重新调整[1]。传统系统常常在设计初始就划定了整体的结构,进行了一体化的设计。因此系统各功能间紧密结合,代码的效率很高。但是由于系统各部分的紧耦合,使得系统维护和升级的成本高、风险大[2]。J2EE和ASP.NET的应用使面向服务架构(Service Oriented Architecture, SOA)的系统得以实现。SOA系统通过简单通信协议(Simple Object Access Protocol, SOAP)把具有独立接口的各个服务模块整合起来以实现一个新的功能。SOA提高了系统的灵活性,同时由于SOAP的特点,其应用范围也不仅仅局限于B/S模式。对于目前已经广泛应用的B/S架构软件系统,建立类似SOA,能处理不同功能模块间灵活组合服务的事务处理模型是本次研究的重点。

1 SOA概述

面向服务架构(Service Oriented Architecture, SOA)系统通过简单对象访问协议(Simple Object Access Protocol, SOAP)将各个接口中立的基本服务模块有效的联系起来,以实现对某一事务的处理。因此,SOA系统可以将相互独立的信息孤岛整合起来,实现新的服务;或可以把系统整体进行划分,封装成一些基本服务模块,以降低各模块间的耦合性,加强模块的复用性。文中主要对于后者进行研究,即系统功能的模块化划分。这种模块化和低耦合性使得SOA系统的更新变得比传统模式下简单。SOA系统中包含三个

收稿日期:2009-01-11;**修回日期**:2009-04-24
基金项目:国家自然科学基金项目(70672097);国家自然科学基金重点项目(70631003)
作者简介:田 冲(1983-),男,山东泰安人,硕士研究生,研究方向为企业管理信息化;李兴国,教授,研究方向为数据挖掘、管理信息系统。

基本角色[3],即:服务请求者(Service Applicant, SA)、服务提供者(Service Provider, SP)和服务注册中心(Service Register Center, SRC)。Web 服务作为实现 SOA 的最佳技术方案之一,服务提供者实现在 Internet 上提供服务,现实中即为 Web Service 模块;服务请求者是 Web 服务的消费者,查询并使用 Web 服务;服务注册中心存储有 Web Service 的目录,可供开发者发布服务或寻找已有的服务。

2　事务处理模型设计

当前主流的 Web 系统开发语言有 ASP. NET、JSP、ASP、PHP、Perl 等。为了使文中的研究具有普遍性,这里不对开发语言进行具体限定,实验环境为:Windows 2003 Server、IIS6.0、固定虚拟目录。

在这里引入事务的概念。每个事务对应着一项服务,而一项服务由单个或多个功能模块(Service Unit)组成。根据面向服务架构的特点,把 Web 系统的角色划分为服务请求者、服务提供者和服务注册中心。为了协调三个角色的活动,这里再引入虚拟处理中心(Virtual Central Processing Unit, VCPU)和用户交互中心(User Interface Center, UIC)。

其中:虚拟处理中心(VCPU)负责接收用户的业务信息,根据业务信息搜索所需的服务并将结果返回给用户交互中心;用户交互中心(UIC)向用户提供交互界面,记录用户的活动状态。该系统的体系层次划分如图 1 所示。

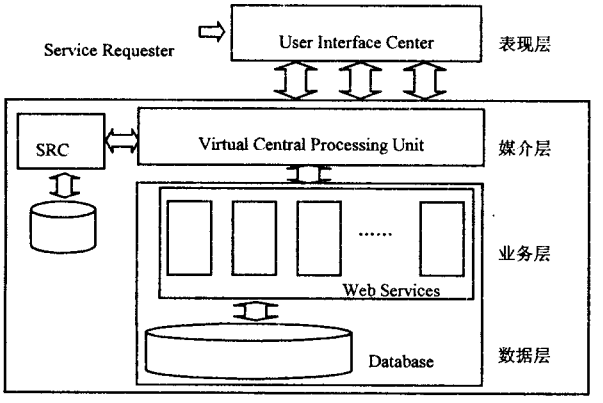

图 1　系统层次划分示意图

事务处理流程如图 2 所示。先在 SRC 注册服务单元(Service Unit)信息,然后把各相关服务单元有序组合,以实现某项服务。用户在交互中心(UIC)激活该服务后,系统获得该项服务的编号(Service ID)并将 Service ID 值传递给虚拟处理中心(VCPU)。VCPU 根据 Service ID 到服务注册中心(SRC)查询服务详细信息。若成功获取该服务的信息,则 VCPU 为该服务创建一条进程信息。VCPU 根据服务单元间的组合信息,依次向用户调用各个服务单元,并收集用户的操作信息。当用户完成各服务单元的操作后,事务的处理过程结束。向用户提示声明信息后,VCPU 将该服务的进程注销。

若用户中途放弃对该事务处理,则 VCPU 根据收集到的用户操作信息,对当前的活动实施补偿操作。而后将该进程信息存储到用户进程表中,供用户再次使用。

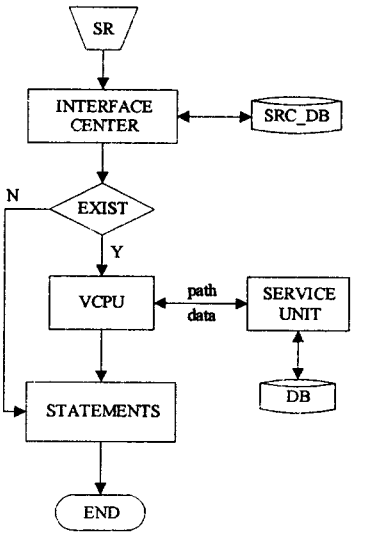

图 2　事务处理流程简图

3　事务处理模型的技术实现

3.1　服务注册中心

服务注册中心提供服务信息、实现新服务的注册和已有服务的检索。因此核心的数据表包括服务单元信息表、服务组合表。

各服务单元在服务注册中心注册后,相互间功能是独立的。一个完整的服务会由一个或多个服务单元组成,即需要确定服务的域。对于由一个服务单元构成的服务,称为原子事务;对于由多个服务单元组成的事务,称为复合事务。服务单元间的组合关系符号设计见表 1。

表 1　关系符号含义

运算符	优先级	说　明
()	1	提高括号内运算的优先级
>	2	从左向右顺序执行
&	3	与运算
||	3	或运算

例如:现有服务单元 A、B、C、D。事务 I 由 A>(B&C)>D 组成,则事务执行过程的服务单元的顺序见图 3。其中的数字表示该服务单元对应的执行序号。

服务单元的执行结果正常则返回值为 True。若执行结果为 False,即该单元操作中有不满足条件的情况,则该事务进程结束,退出执行。由图 3 可知,I 事务的实现由 A、B、C、D 四个服务单元组合而成。流程先从 A 服务单元开始。A 服务操作完成后,由于提升了 B&C 的优先级,则进行 B、C 两个服务单元操作,并将执行结果取"&"操作。若 B、C 两个服务单元中的操作结果都为 True,则再进行 D 的操作。当 D 执行结束后,整个事务 I 的处理过程结束。

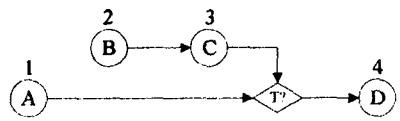

图 3 事务 I 执行流程示意简图

流程控制的实现方式为:只显示当前将要进行的服务单元,当前服务操作完成后才显示后一步的服务。用户未完成前序步骤的情况下,后续服务单元对用户是不可见的。

3.2 服务单元

与常见的 Web 服务开发相类似,创建独立的服务单元以实现某项功能,而后将该服务单元放置在服务器的主虚拟目录下。在服务注册中心通过填写相应信息,注册该服务单元。注册后与其他服务单元组合以实现新的功能,即:定义一个新的服务。

服务单元描述规范的统一是实现服务间灵活组合的前提[4]。根据 WSDL 的 Schema,Web 服务的描述包括:服务名、操作参数、输入、输出和服务属性简介五元组[5],即:

SU={ServiceName,Op,Input,Output,Profile}

引入 RelatedService = {PreService, SubService},(即:前向服务集与后向服务集)到服务单元的描述中,以满足本研究中事务处理模型流程控制的需要。因此服务单元描述扩展成为一个包含相关服务项(Related-Service)的六元组。

由于本模型对每个事务的处理都要建立对应的进程,因此服务粒度对服务器的性能有很大的影响。粒度过细与服务器操作频繁,使服务器负担过大。粒度过大会使模块的通用性低、系统灵活性受到限制[6]。考虑以上特点,把服务单元的粒度定义限定在较大的粒度上,以降低服务器负荷要求,强调对事务的总体流程控制。

3.3 用户交互中心

用户交互中心作为系统的表现层,为用户提供操作界面。用户交互中心把系统的各个服务单元整合到同一框架下,把各项服务通过列表的形式列取出来,并提供索引功能,方便用户查询。用户交互中心主要实现验证用户登录、记录用户操作信息、对业务进程的一些关键数据临时备份。因此用户交互中心的核心数据表格包括用户信息表(User Info)、进程信息表(Process Info)和临时信息表(Tem Info)。

3.4 虚拟处理中心

用户在交互中心激活某一服务后,系统相应产生一个事务。VCPU 获取该事务对应服务的编号,为该事务创建一个进程。然后 VCPU 根据编号查询服务的详细信息。首先,VCPU 在用户交互中心查询该服务是否存在于该用户的进程表中。若存在,则取出该进程中的服务信息,继续执行剩余步骤。若不存在则访问服务注册中心(SRC),搜索对应服务。搜索该服务不存在时提示用户没有此项服务。若存在,则将服务信息存储到用户进程表中。然后 VCPU 把得到的信息进行处理、解析,依次执行各个服务单元。

(1)相关参数的定义。

与用户相关的参数定义:用户编号(User Id)、用户权限(Power)。用户登录交互中心后此参数便可获得;

与进程相关的参数定义:进程编号(Process Id)、进程状态(Process State)、计数器(iCounter);

与事务相关的参数定义:服务编号(Service(Id))、服务单元集合(Service(Tree))、权限集合(Service(Power))、状态集合(Service(State))。

(2)业务进程的初始化。

首先初始化业务的进程信息(包括:Process Id, Process State, iCounter, Service(Id), Service(Tree), Service(Power), Service(State)),获取业务编号后赋值给 Service(Id)。根据 Service(Id)到用户交互中心(UIC)的用户业务列表中查询该用户下是否存在此业务。若存在,则把业务执行的信息分别赋值给 Service(Tree)、Service(State);若不存在,则到服务注册中心(SRC)查询该业务对应的服务信息赋值给业务进程的各参数;若查询结果不存在,则返回查询失败信息。

(3)解析服务单元组合信息。

服务单元的组合信息以字符串的形式存储于 Service(Tree)中。Service(State)则是为 VCPU 控制执行过程而记录的辅助信息。因此 VCPU 要执行各个服务单元,还需把字符串其中的组合信息解析出来,转换成实际的操作流程[7]。设定集合 T,S 和 P。

$\Phi(T)=\{t_1,t_2,\cdots,t_n\}$,服务 T 包含的服务单元集合;

$\Phi(S)=\{s_1,s_2,\cdots,s_n\}$,各服务单元对应状态集合,$S$ 对应取值集合为{Null,True,False,Abandon}

$\Phi(P)=\{p_1,p_2,\cdots,p_n\}$,各服务单元对应权限集合。

(4) 遍历执行过程。

根据 $\Phi(S)$ 中的信息获取进程起点。服务单元 t_i 状态对应 s_i 值为 True 时则该服务单元已完成,为 Null 则未完成。代码如下:

```
//获取进程起点
i=1;
iCounter=0;
For(i<n+1){
    If(si==Null){
      iCounter=i;
      Exit For;
    }
    Else{
      i++;
    }//End If
}//End For
...
//事务流程控制模块(访问 Service Unit);
获取状态参数;
If(返回状态为 Abandon){
保存进程状态信息至 Process Info 表;
则注销本进程;
}
ElseIf(返回状态为 True){
    iCounter++;
    SRC 中查询服务单元 tiCounter;
    获取 tiCounter 的操作权限 piCounter;
    If(Power==piCounter){
      显示服务单元 tiCounter;
    }
    Else{
      提示无权访问;
      注销进程;
    }End If
  }
ElseIf(返回状态为 False){
    Show Error;
  }//End If
```

(5)异常处理。

异常类型包括:超时、越权、放弃、冲突。

①超时处理:当用户长时间无操作,则在注销 SESSION 变量的同时,在用户交互中心的进程信息表中保留用户当前进程状态信息,以便用户下次调用。

②越权处理:在进程执行的过程中核对用户权限信息,若不满足该服务单元的权限则禁止用户访问,给出提示信息后结束进程。

③放弃处理:用户主动放弃某服务时,则清除用户交互中心该服务的进程信息,然后在服务器中注销该进程并把该用户设置为闲置状态。

④冲突处理:同一用户的重复登录会造成冲突。对于此类冲突,通过设置用户状态的形式来解决。当一个用户登录后,把该用户的 Status 标注为 ONLINE,则该用户不能再次登录。用户注销退出系统时则标注用户的 Status 为 OFFLINE。

4　某精品课程系统的实例应用

本精品课程系统开发所用的 Web 语言为 ASP (Active Server Pages)。系统的运行环境为:Windows 2003 Server、IIS6.0、固定虚拟目录。根据传统模式下 Web 教育系统的应用现状[8]和已建立的原精品课程系统的使用,我们发现以下不足:

1)系统功能结构僵化,无法适应教学要求的变化;

2)缺乏对课程系统和学习流程的总体控制,使新的需求不断增加;

3)系统功能的复用性比较低,出现类似功能间的重复建设,增加了代码量。

整个精品课程系统的功能单元划分为:学员注册(A)、课程学习(B)、考试模块(C)、成绩管理(D)和证书管理模块(E)。学员的学习的应用流程为:学员在课程网站注册后在线学习课程,学习完毕后参加课程考试,考试合格并领取证书后,整个流程结束。

但是当学员达到一定数量后,系统需要引入学员管理模块(F)以管理各考点的学员信息。还需要引入学习管理模块(G)以细分学习类型。采用第 3 节中提到的事务的处理方法,以模块化事务模型设计的课程系统则可以把流程重整为如图 4 所示,图中的(a)可以分解成(b)和(c)两个流程。

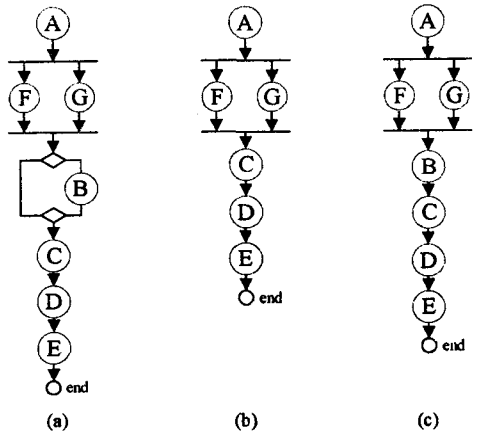

图 4　重整后的流程示意简图

在整个事务流程的执行过程中,VCPU 起着中枢的作用。VCPU 的存在使得系统各模块动态、有机地组合在一起。当后期有新功能加入系统时,将该模块

(下转第 69 页)

和 TURN 服务器分配的端口号分别为 8100 和 9100。B 端得到地址后就开始用 STUN 从本地的地址(192.168.151.1:2345)来尝试连接 A 端所提供的地址。B 端的应答消息体中 SDP 地址结构与 A 类似。消息体如下所示:

c = IN IP4 202.195.2.1

m = audio 9100 RTP/AVP 0

plus = 202.195.1.1:8100

以下对 A 分两种 NAT 情况讨论:

(1)若 A 为非对称性 NAT:因为 A 处于非对称性 NAT 之后,A 的 STUN 地址可以连通,那么 A 端送来的 c 和 m 的地址在 B 端就可以不去测了。这时 B 端发应答给 A,同时可以向测通的 A 端的这个地址发送音频流。A 收到应答后,同样从本地地址(192.168.150.1:1234)发送 STUN 请求来尝试连接 B 应答中所给出的地址直到有一个可以连通为止。同样的如果 B 端在非对称型 NAT 之后,那么 B 的 STUN 地址将可以通信。

(2)若 A 为对称型 NAT:首先 B 端也会去尝试连接 A 端的 STUN 地址,B 端的 STUN 消息到达 A 端的 NAT,因为 A 端是对称型 NAT,这个包会被丢掉,A 没有应答,那么 STUN 地址将不会通。这个时候 B 端会测 A 的 TURN 地址。B 端连接 A 端的 TURN 地址会成功。B 端在收到 A 应答时就可以用地址向 A 发送语音流,连通成功。

该节主要讲述的是媒体流数据的 NAT 穿透。进行媒体数据传输的前提是已经建立信令连接,关于 SIP 信令的穿透也可以用类似的方法处理。

5　结束语

文中分析了应用 P2PSIP 系统时面临的 NAT 穿越的问题,在分析各种 NAT 穿越方法的基础上提出一种 STUN + TURN 方式穿越 NAT 的方案。同时通过对会话描述 SDP 进行扩展(增加头域)以实现该方案。采用该方案的 P2PSIP 系统可以实现正常通信。

参考文献:

[1] Bryan D, Matthews P, Shim E, et al. Concepts and Terminology for Peer to Peer SIP[S]. draft - ietf - p2psip - concepts - 01, 2007.

[2] Rosenberg J, Schulzrinne H, Camarillo G, et al. SIP: Session Initiation Protocol[S]. RFC 3261, 2002.

[3] 何宝宏. 浅析 NAT 的类型[J]. 电信网技术, 2004, 8(8): 427 - 430.

[4] 张永强, 张捍东, 赵金宝. SIP 协议栈研究[J]. 计算机技术与发展, 2007, 17(11): 49 - 51.

[5] 王健婷, 赵　霞, 刘　杰, 等. 基于 P2PSIP 的 NAT 穿透方法的研究[J]. 北京工商大学学报, 2008, 5(5): 615 - 618

[6] Mart P A, Sijben P, Brim S, et al. Middlebox Communications (midcom) Protocol Requirements[S]. RFC3304, 2002.

[7] Rosenberg J, Weinberger J, Huitema C, et al. Simple Traversal of User Datagram Protocol Through Network Address Translators[S]. RFC 3489, 2003.

[8] Rosenberg J, Mahy R, Huitema C. Traversal Using Relay NAT[S]. Internet - Draft, 2005.

(上接第 65 页)

在服务注册中心注册后,即可通过与其他服务模块的组合实现新的功能。

5　结束语

文中把 SOA 模块化整合思想引申到 Web 软件系统的建设过程中。采取与整合相反的方式,把系统进行模块化划分,设计了用户交互中心(UIC)、虚拟处理中心(VCPU)等媒介以协调各个系统各个模块的组合,提高了系统的可扩展性和灵活性,增强了系统的适应能力。同时,本事务处理模型通过引入进程,可以对用户某项业务的使用状态加以监控。后续的工作要把用户的业务使用偏好引入到模型中,在统计用户业务偏好的基础上改进服务组合,对不同的客户提供准确的服务。

参考文献:

[1] Jeng J J, An Lianjun. System Dynamics Modeling for SOA Project Management[J]. Proceedings of the IEEE, 2007(7): 286 - 294.

[2] 杨象驰, 李鹏飞. 基于 SOA 的邮政物流信息系统规划[J]. 计算机工程与设计, 2007, 28(19): 4825 - 4827.

[3] 王一宾, 张玉州, 程一飞. 几种新型软件体系结构风格的分析[J]. 计算机技术与发展, 2008, 18(8): 39 - 42.

[4] Wu Jian, Wu Zhaohui. Similarity - based Web Service Matchmaking[J]. Proceedings of the IEEE, 2007(1): 287 - 294.

[5] 周　煜, 成　斌, 童维勤. 基于 QoS - Ontology 的 Web 服务选择 Broker 模型[J]. 计算机应用与软件, 2008, 25(9): 10 - 16.

[6] Chang Fu - chun, Hung Tai - chang, Chiou Young - jang, et al. Design and implementation of web service integration Tool [J]. Proceedings of the IEEE, 2005(10): 91 - 96.

[7] 李　磊, 牛春雷, 陈宁江, 等. 一种高效的 Web 服务性能优化策略[J]. 计算机研究与发展, 2007, 44(7): 1191 - 1198.

[8] 惠敏顺, 朱国进. 基于 SOA 的分布式程序设计竞赛系统的研究[J]. 计算机技术与发展, 2008, 18(10): 123 - 126.