一种消息驱动的 SOA 系统集成方法①

张功萱, 陈 瀚, 王永利, 王 辉

(南京理工大学 计算机学院, 南京 210094)

摘 要：针对企业 SOA 平台异构系统和服务的集成问题, 提出了一种以 Web 服务为封装形式, 消息队列为联系机制, 软件管道为并行计算方案的的支持交互方式扩展的高性能松散耦合 SOA 系统集成方法。在应用层结合 RPC 原理、滑动窗口机制, 引入商业并行计算软件管道技术, 以消息驱动的方式实现了平台各系统间的服务同步/异步调用、并行事务处理以及拥塞控制, 满足了业务交互和共享的需求。

关键词：面向服务架构; Web 服务; 消息驱动; 软件管道; 异步 RPC; 拥塞控制

Message-Driven SOA Integration Approach

ZHANG Gong-Xuan, CHEN Han, WANG Yong-Li, WANG Hui

(Dept. of Computer Sci. & Tech, Nanjing University of Sci. & Tech., Nanjing 210094, China)

Abstract: Aiming at the issues of interaction among heterogeneous systems in Service-Oriented Architecture platfrom, this paper proposed a loose-couple business integration model in the form of message-driven Webservices which support multiple interactive modes. Consider the demand of service quality for integrated applications, this paper introduced software pipelines for parallel computing and also elaborated on synchronous/asynchronous calling, transaction and congestion control of Webservices based on the approach of remote procedure calling and sliding window mechanism that operates in an equipment reservation system.

Key words: SOA; web service; message-driven; software pipelines; asynchronous RPC; congestion control

面向服务架构(Service-Oriented Architecture, SOA)系统集成方法主要包括企业应用集成(Enterprise Application Integration,EAI)和企业服务总线(Enterprise Service Bus, ESB)等技术。目前对 EAI 和 ESB 的研究主要集中在异构应用接口标准化和组件化构建方面, 如 Pan 等提出了一种集合制造、过程控制、订单、支付、物流等各种服务的基于 ESB 的销售系统模型, 其研究内容之一是异构应用的数据交互和集成访问[1], 另一些研究从底层数据整合角度提出“把分布在异地的异构数据库源通过 Web 服务连接起来形成一个异构的中心数据库并提供透明的用户接口”[2]的方案。

本文结合传统 EAI 的中间件技术在业务数据整合方面的优势, 以及下一代 ESB 的接口标准化、组件化技术的开发便利, 在中国科学院仪器设备共享平台开发过程中, 实现了一种基于消息中间件(Message Oriented Middleware, MOM)技术的 SOA 系统集成技术(第 1 节), 联系实际重点阐述了以消息为联系机制、管道为运行载体的高性能并行服务整合方案, 并研究了可靠性和质量保证方法(第 2 节), 以应对传统 EAI 集中式通信方式面临的高负载和稳定性问题, 同时灵活定义便于扩展的 XML 格式服务调用指令, 避免了 ESB 异构服务之间过于独立、分散而导致的资源浪费, 为分布式企业应用平台的构建提供了一种途径。

1 系统结构及相关技术

本方法实现的仪器设备共享平台由 Web 预约服务和终端服务 2 个子系统组成。Web 预约服务系统提供全院设备的采购、管理、预约、统计缴费功能, 各所

① 基金项目:国家自然科学基金(60803001)

收稿时间:2011-06-23;收到修改稿时间:2011-07-23

级单位使用相互独立的终端服务系统为用户提供登录和监控、设备使用、计费等功能。

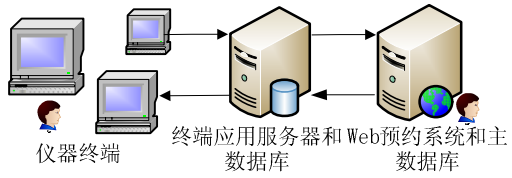

图 1 应用平台示意图

终端应用服务器(.NET 平台)是连接客户端与本地数据库服务器(SQLServer)和远程 Web 服务器的桥梁，它一方面以同步或异步的方式从 Web 预约服务系统(J2EE 平台)主数据库(Oracle)获取设备和用户授权、预约信息到本地数据库，一方面响应客户端登录、注销请求，向本地数据库中写入工作日志、计费信息，再将这些信息回传到与 Web 预约系统相连的主数据库。

用户可以通过浏览器访问全院统一的 Web 预约系统并发出预约，然后在本单位仪器终端登录进行相应操作。该过程中的所有系统间业务交互采用消息驱动的 SOA 服务集成方法，其总体结构如图 2 所示。

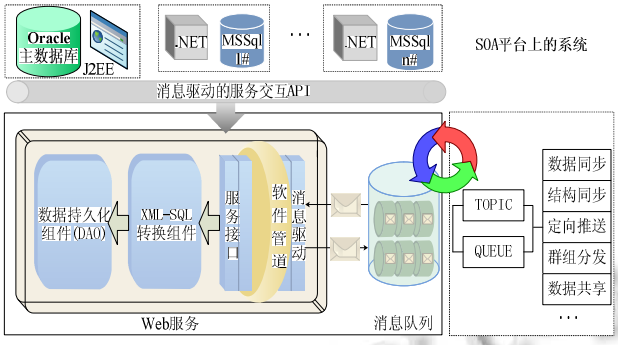

图 2 消息驱动的 SOA 服务集成模型

1.1 消息队列

消息队列（Message queue）是一种进程间或线程间的通信方式。消息队列通过为不同环境的生产者/消费者进程维持一个消息队列来实现各方的异步通信。也就是说，消息的发送方(生产者)和接收方(消费者)不需要同时与消息队列互交，消息会保存在队列中，直到接收方取回它。消息队列使用 Queue(队列)和 Topic(话题)两种方式实现不同的语义：Queue 方式的一条消息仅能被一个消费者收到，如果在消息发送时没有可用消费者，那么它将被保存在队列中等待可用消费者；Topic 方式的一条消息发布(publish)时，它将发送多个拷贝到所有感兴趣的订阅者(subscriber)[3]。

1.2 消息结构

在 J2EE 平台上，JMS(Java Message Service)消息是消息队列中信息处理的基本单元。消息结构如图 3 所示，其中消息头(Headers)定义了消息发送和接受的方式、消息 ID、消息邮戳和过期时间、消息目的地、多个消息之间关联等；消息属性(Properties)则扩展了消息头的功能，提供了区分发送进程和接收进程标识的 JMSXAppID 和 JMSXConsumerTXID、定义消息组和组内消息顺序的 JMSXGroupID 和 JMSXGroupSeq 等；消息有效载荷(Payload)是由用户自定义的消息内容。

1.3 XML 交互指令和内容的定义

消息队列可作为一个轻量级的交互工具，通过消息中间件 ActiveMQ 进行.NET 和 J2EE 平台的交互。事务都通过指令进行交互，指令以 XML 格式存放在 JMS 消息的有效载荷(Payload 段)中完成传递。

由于目前的关系数据库中并没有直接执行 XML 调用的统一标准，XML 指令必须先转换为 SQL 语句，在.NET 平台下可以通过 LINQ 实现，J2EE 平台并没有同样强大的库进行该操作，一种方法是基于 DTD(文档类型定义)[4]实现 XML 到 SQL 的映射。以下给出一套简洁满足应用平台需求的 XML 交互格式定义。

图 3 JMS 消息结构

如下 XML 语句描述表的创建和结构变更:

```
<msg>
  <op>创建(Create)或变更(Alter)</op>
  <!--需要创建或更改的表字段和类型. -->
  <tab name = "t1">
    <field name = "f1">字段类型</field>
    <field name = "f2">字段类型</field>
  </tab>
```

```
…
<!--表的外键约束和唯一性约束等. -->
<constraints>
  <ref FK_ t1_ t2= " t1_id"> t2_id</ref>
  <unique tab = "t1">f1_ f2</unique>
</constraints >
</msg>
```

对于数据的增删改使用如下 XML 语句进行描述:

```
<msg>
  <id>单位编号</id>
  <tab>表名</tab>
  <op>操作类型:增(i)删(d)改(u)</op>
  <!--需要操作的记录对应的主键内容. -->
  <keyid>
    <keyid>主键 id1</keyid>
    <keyid>主键 id2</keyid>
    ...
  </keyid>
</msg>
```

以上 XML 语句主要功能是将本地数据变更情况通知给目标数据库，只描述被变更记录的键 ID 等信息，实际数据的获取则由与目标数据库相关的应用程序调用 Web 服务取得数据库记录后在本地数据库中完成同步，数据库记录可用 XML 标签表示各字段形成数据表 XML 文档。

1.4 软件管道的引入

本文采用基于 JMS 的第三方消息中间件 Apache ActiveMQ 实现消息队列服务，一个消息服务器中可定义多个消息队列并对其中的消息分类筛选以形成面向服务的交互队列或话题，从而在消息层面初步划分业务，如表 1 所示。

表 1　事务消息分类和管道命名

消息分类表达式	管道名称
UnitID > 001 and UnitID <= 100	P_001_ti_100
UnitID > 101 and UnitID <= 200	P_101_ti_200
UnitID > 201 and UnitID <= 300	P_201_ti_300

为了对划分后的业务逻辑进行负载均衡处理和并行计算，引入软件管道(Software Pipeline)架构，管道是一种以控制顺序的方式对业务流程的离散任务进行调用的虚拟执行设备[5]。由于商业应用处理事务并有严格的 FIFO 顺序，传统并行计算方法(如多线程、SMP 和集群)缺乏必要的灵活性，而在多核处理器上使用多管道可以实现商业应用的并行计算。同时配合数据库分片(Database Sharding)技术，可对集中资源进行负载分配，如图 4 所示。

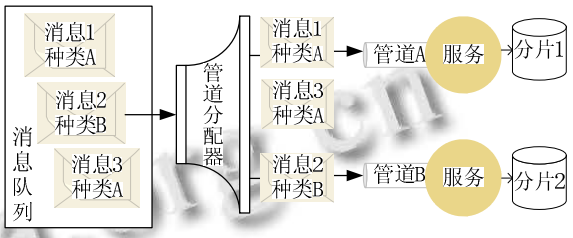

图 4　消息队列和管道

1.5 消息驱动的工作过程

消息驱动的具体工作过程为：

① 发送方设置消息的头、属性和有效载荷并发往指定队列。

② 接收方指定其关注的 Topic 或 Queue 名称，并限定其收听的消息“频道”，对消息进行过滤筛选。

③ 接收方接收消息，将有效载荷部分解析到 XML 树，消息驱动将业务操作指令输送到软件管道。

④ 软件管道中的管道分配器将业务操作路由至不同的管道，如果相应管道不存在，分配器动态分配一个管道。

⑤ 管道调用 XML-SQL 转换组件将 XML 指令转换为 SQL 语句，根据业务需求调用 Web 服务进行数据持久化操作。

一般情况下，发送方在消息成功发送到队列后不需要关心消息何时、如何传递。接收方与队列建立连接后，对消息的获取有主动获取和监听两种方式。前者通过指定需要的消息条数进行接收，类似于登录邮箱收取邮件的动作，是完全异步工作方式，后者需要向系统注册一个 MessageListener 接口进行消息监听，在 JavaBean 编程中称为 Message-Driven Bean. 通过 ActiveMQ 提供的 CMS 编程接口，同样可以在.NET 环境中使用 C++或 C#编写消息驱动。该接口接收到消息后，转入消息处理过程。

```
public class MessageProcessor
implements MessageDrivenBean, MessageListener
{
```

```
public void onMessage( Message msg ) {
   if( msg != null ) {
      TextMessage txtMsg = msg;
      processMsg( txtMsg ); }
}
private void processMsg (TextMessage txtMsg)
{// 消息处理过程(与管道对接)}
}
```

应用平台中表结构和数据的同步和增量更新，包括预约信息的定向推送，任务招标的群组分发，区域中心各单位间的子数据库设备信息的共享等都可通过消息实现。

例如用户在 Web 预约系统上预约了某台设备，通过审核后发送消息到交互队列，终端服务器收到消息后解析消息 XML 内容表示的数据操作，通过 Web 服务接口从主数据库获取记录后插入本地数据库，设备终端即可绑定该预约信息；又如招标方在预约系统上发布的信息可以分发到相应的交互话题中，相关的终端服务器均可获得该招标信息并回应；兄弟单位间有时需要进行数据的共享，但是必须保证本系统内数据的安全性，比如 A 单位可以向 B 单位提起设备预约申请，但是 A 单位终端服务器无权获取 B 单位的预约信息，此时可以向共享队列中发送包含 A 单位预约人员的 IC 卡信息、受理方(B 单位)设备编号等信息的消息，B 单位终端服务器接收到消息并验证权限后安排将用户信息插入本单位数据表中，A 单位人员即可登录 B 单位终端系统。

2 系统的服务质量

上节实现了基于消息队列的基本业务交互，本节将探讨并提出消息交互过程中 Web 服务质量和可靠性保证方法。

2.1 消息控制下的 Web 服务异步调用

相关研究提出了一种基于 WSE 和 SOAP 消息队列的 Web 服务异步调用方法[6]，通过向微软消息队列(MSMQ)中发送 SOAP 消息实现远程调用(RPC)的异步发送/接收，该方法基于微软消息队列，依靠 SOAP 消息中的确认地址返回调用结果，涉及到请求和响应关联问题，发送方和接受方需要分别维持一个队列。

在本应用平台中，调用端(发送方)和服务端(接收方)共享同一队列中的 Topic，调用端用上文定义的 XML 指令格式生成调用请求消息，设置消息的属性和 JMSCorrelationID 并发送，消息队列将该消息持久化并返回确认，此时调用端回到主线程，服务端的消息驱动接到消息后执行相应操作并返回关联调用结果(使用与调用端相同的 JMSCorrelationID)到消息队列，此时服务端回到主线程，消息驱动接收到返回消息通知调用端调用完成。过程如图 5 所示。

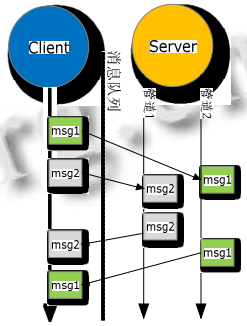

图 5 基于消息队列的 Web 服务异步调用

由于调用端和服务端向消息服务器注册了持久化订阅 ID，所以调用和返回消息会先保存到消息服务器，即使双方不同时在线也可以完成 RPC。

2.2 消息控制下的 Web 服务同步调用及拥塞控制

目前对 Web 服务同步调用的方法主要是进行应答式的交互[7]，对于调用不成功的情况要求调用端重新调用。但是对于网络堵塞原因造成的调用失败，如果重复发送调用请求则会导致问题。例如终端系统需要向主数据库发送设备工作记录，Web 服务将记录插入一个具有自增 ID 的数据表 DB_LOG 中，假如在连接中断过程中终端进行了多次调用，则连接重新建立时，Socket 缓冲区的所有超时调用请求会同时进行数据库插入操作，导致多条重复记录。而且重复调用的方法也增加了 Web 服务的负担，研究证明 Web 服务在单位时间内处理能力随并发请求数的增加而下降[8]。

由于管道的引入，Web 服务调用被分配到某个命名管道，如果出现重复调用，管道分配器会检查是否已存在同名的管道，若存在则忽略重复请求，避免多次调用导致的记录重复和服务资源浪费问题。同时，管道的引入增强了系统处理负载的能力，如图 6 所示，将应用平台中的“预约审核”(服务 A)和“结果发放”(服务 B)2 个服务管道化后，整个预约流程的性能因此提升(单位为 TPS，事务数/秒)。

由于商业计算负载的不确定性，无法预先判定拥

塞情况，本文采用在消息队列中建立滑动窗口的方式进行 Web 服务拥塞控制。对于不限定接收顺序的 Web 服务请求，只需要调用端和服务端共享一个限制容纳消息数目的 Queue，当 Queue 满时，不能再发送请求消息，必须等待服务端接收到一条消息并完成 Web 服务操作后确认(ACK)该条消息，Queue 中才会清除该消息，此时可以继续发送请求。对于限定接收顺序的批量 Web 服务请求(设请求数为 n), Web 服务调用端和服务端各维持一个 Queue，并限定容纳的消息数目分别为 x(x<n)和 1，服务端每处理完一个请求并 ACK，此时调用端才能发送下一个请求。图 7 显示了发送窗口尺寸为 2，接收窗口尺寸为 1 时的情况。

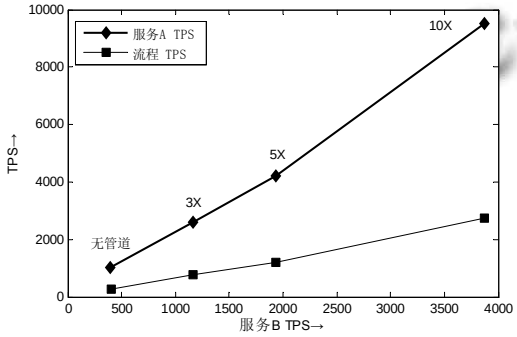

图 6　系统管道化后事务处理能力的提升

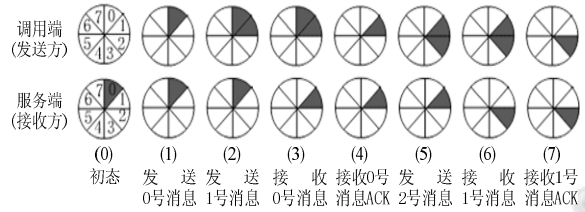

图 7　Web 服务调用的同步和拥塞控制

Web 服务端根据本地负载轻重程度动态地增加或减小 Queue 的尺寸；调用端根据消息服务器返回的消息接收情况也可判断服务端的处理能力，实时调整发送调用消息的速度，实现了调用者对 Web 服务负载情况的掌握。

2.3 Web 服务事务消息处理

DBMS 要求数据库事务拥有 ACID 特性，即原子性(Atomic)、一致性(Consistency)、隔离性(Isolation)和持久性(Durability)，对于异构平台的事务处理，一种方法是在独立的中间数据库建立全局的临时事务表，由第三方不断监视该表中记录变化并根据时间戳进行事务处理[9]，然而该方法必须由分散在 ESB 系统中的各个客户端自行查询事务处理结果，影响了 Web 服务的性能。

在图 2 所示模型中，一致性和持久性由数据持久化组件和底层关系数据库保证，而涉及到并发 Web 服务请求以及有次序的 Web 服务调用时，需要保证数据库操作序列的原子性和隔离性，必须对上文提出的 Web 服务异步/同步调用方法加以改进。

J2EE 开发中可通过 createQueueSession 创建 JMS 事务会话，.NET 平台通过为 ActiveMQ 消息队列中间件 createSession 函数指定 SESSION_TRANSACTED 参数创建事务消息会话，为消息设置同样的组号(JMSXGroupID)后将事务消息分配到同一个消息组(MessageGroups)中并按组内顺序编号(JMSXGroupSeq)，保证事务消息一次性被传递到接收方，如果其中任一条消息失败，接收方将不能接到任何消息，由客户端重新接收或发送方重新发送。接收方在收到消息后，根据 JMSXGroupSeq 顺序进行 Web 服务事务操作，如果其中某个操作失败则撤消之前的所有操作，并取消之后的所有操作，回滚到初始状态后返回失败消息给事务调用者。对于接收方存在多个监听线程的情况，消息队列客户端的排他消费者(Exclusive Consumer)方式支持接收方存在多个普通的或排他的监听线程，当发送事务消息时，只有一个处于激活态的排他消费者可以接收全部的 Queue 消息，如果该 Consumer 中断，则激活另一个消费者来接收全部消息(包括已经被前一个消费者接收过的)，此举保证了事务队列的线程安全性。

3　结语

本文结合消息队列的高级特性、软件管道等最新技术，提出了消息驱动的工作流、数据流和应用服务的具体集成方法，具有良好的通用性和可扩展性，从实际应用出发解决了 SOA 集成环境下存在的 Web 服务同步/异步调用、可靠传输和拥塞控制等问题，保证了业务质量。

参考文献

1 Xiulong Pan, Wei Pan, Xiao Cong. SOA-based Enterprise Application Integration. 2010 2nd International Conference on Computer Engineering and Technology, 2010,7:564–568.

(下转第 160 页)

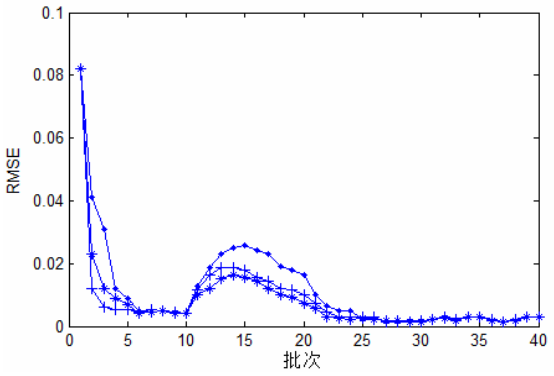

图 3　跟踪误差收敛图

模型参数发生变化通过改变批次间的参数来仿真。假设参数 T_2 在第 10 到第 20 个批次间发生变化，每个批次结束后增加 0.5。将模型参数 G 不变时和 LTVP 模型下的迭代学习算法进行比较，如图 3 所示，“.”“+”“*”三条曲线分别表示 G 不变、RLS 方法辨识参数和 FFRLS 方法辨识参数下的跟踪误差。可以看出，在第 10 个批次后开始变化时，跟踪误差能保持在一个较小的值内，在第 20 个批次后，模型参数停止变化，三种情况的跟踪误差都开始收敛，但 FFRLS 能更准确的辨识 LTVP 模型，所以它的跟踪误差在收敛过程中最小。原因是模型参数发生变化时，遗忘因子能大大减小以前的“错误”数据对算法的影响，具有更高的自适应性。

5　结论

本文将迭代学习控制方法应用于黄酒发酵过程的温度控制中，用于解决过程中模型参数变化的控制问题。学习律中的参数由 FFRLS 方法辨识得到，仿真结，果证明了算法的有效性，并且与 RLS 方法进行了比较在模型参数发生变化的情况下，FFRLS 方法比 RLS 方法有更高的自适应性。

本项目组研发的黄酒发酵自动化智能控制系统已在浙江古越龙山绍兴酒股份有限公司投入使用，运行稳定可靠，并且在 2011 年 4 月通过了教育部鉴定。

参考文献

1 孙明轩,黄宝健.迭代学习控制.北京:国防工业出版社,1999.

2 于少娟,齐向东,吴聚华.迭代学习控制理论及应用.北京:机械工业出版社,2005:32.

3 Campi MC, Sugie T, Sakai F. An Iterative Identification Method for Linear Continuous-Time Systems. Automatic Control, IEEE Trans. on, 2008,53(7):1661–1669.

4 Eksioglu EM, Tanc AK. RLS Algorithm With Convex Regularization. Signal Processing Letters, IEEE, 2011,18(8): 470–473.

5 Yixin Xu, Zhihua Xiong. An Optimal Adaptive Iterative Learning Control in Nonlinear Systems Using Perturbation Model. Proc. of the International Conference on Information Computing and Automation(ICICA'2007). Chengdu, China, Dec 20-22, 2007,1:11–14.

6 Xiong Zhi-huang, Zhang Jie, Dong Jin. Optimal Iterative Learning Control for Batch Processes Based on Linear Time-varying Perturbation Model. Chinese Journal of Chemical Engineering, 2008,16(2):235–240.

7 丁锋,萧德云,丁韬.时变系统遗忘因子最小二乘法的有界性收敛性.控制理论与应用,2002,19(3):423–427.

(上接第 90 页)

2 王玉标,文俊浩,赵瑞锋,饶锡如.基于 Web 服务的异构数据库共享及同步机制.计算机工程与设计,2009,30(24):5774–5777.

3 Bruce Snyder, Dejan Bosanac, Rob Davies. Active MQ In Action. Manning Publications, 2011:23–48.

4 Pensri Amornsinlaphachai. Translating XML Update Language into SQL. Journal of Computing and Information Technology, 2006,2:81–100.

5 Cory Isaacson. 多核应用架构关键技术——软件管道与 SOA.吴众欣译,北京:机械工业出版社,2010.

6 刘大玮,刘瑞虹.基于 WSE 和消息队列的异步 Web 服务研究及实现.计算机工程,2010,33(8):127–129.

7 Baoan Li. Research and Application of SOA Standards in the Integration on Web Services. 2010 Second International Workshop on Education Technology and Computer Science, 2010,199:492–495.

8 周晓强.Web 服务异步调用模型的研究与实现.武汉理工大学,2009.

9 Feifei Tao. Research of Cross-platform Intersystem Integration Technology Based on SOA. 2010 International Conference on Future Information Technology and Management Engineering, 2010,1:386–389.