文章编号:1001 - 9081(2005)02 - 0446 - 03

基于 Proxy 模式的 AMI 模型优化方法

何智华[1],邹北骥[2]
(1. 湖南大学 软件学院,湖南 长沙 410082;2. 湖南大学 计算机与通信学院,湖南 长沙 410082)
(hzhhua@sina.com)

摘 要:消息通信是企业应用集成的关键。为了改善消息的灵活性、重用性和扩展性等问题,提出一种基于 Proxy 模式的 AMI 消息通信模型。该模型保留了 AMI 中原有的 Callback 与 Polling 的核心算法,对原 AMI 模型进行了耦合分解,优化了回调技术,使得 AMI 更能满足企业集成中消息的扩展与重用等需要。最后给出了模型的设计实现以及项目中的模拟测试数据。实践表明,该模型能较好地满足企业应用集成中的业务需要。

关键词:代理;异步;异步方式调用;回调;扩展标记语言
中图分类号:TP311 **文献标识码**:A

Optimizing AMI model of CORBA based on the proxy pattern

HE Zhi-hua[1], ZOU Bei-ji[2]
(1. *Institute of Software, Hunan University, Changsha Hunan* 410082, *China;*
2. *Institute of Computer and Communication, Hunan University, Changsha Hunan* 410082, *China*)

Abstract: The message communication is the key of EAI. In order to improve the facility, reuse and expansibility of message, this paper put forward a communication model based on proxy pattern, which reserved the core arithmetic of callback and polling, decomposed the coupling of the original AMI model and optimized the technology of callback, which made more satisfy the need of expansibility and reuse in EAI. Here, the ways to design and implementation, the simulative testing datum in project were introduced. The result shows that the model can be preferably satisfy the need of business in EAI.

Key words: Proxy; asynchronism; AMI; Callback; XML

0 引言

在企业应用集成(EAI)中,CORBA[1]作为分布式应用总线,其中的消息服务[2]是 CORBA 系统的核心服务之一,是实现系统各个部件之间协同和资源共享的基础。

在 CORBA 的消息服务中,具体的消息通信模型主要分为同步方式调用(Synchronous Method Invocation,SMI)和异步方式调用(Asynchronous Method Invocation,AMI)[2]。企业的业务需求通常要求应用程序之间能以一种异步非耦合的方式进行通信,并对数据传输的可靠性与效率要求较高。因此,异步通信方式在企业集成中被广为采纳。

AMI 模型采用回调(Callback)与轮询(Polling)的方式,在客户端实现异步调用,服务器端同步处理,对异步通信提供了一定的支持。目前,国外对 AMI 研究较多的是开源 CORBA 的 ORB,它是面向高性能、实时最小化整合设计的通信模型[3];在国内,主要有基于 StarBus 平台并遵循 CORBA 规范的消息回调服务模型[4]。但是两者在目前来说还不完善,如:对象间的通信基本上是点对点通信,无法做到一对多、多对多;对象间的通信仍然是一种耦合方式;客户端无法获知回调对象的信息是否已经丢失,等等。

因此,需要对 AMI 模型在通用性、扩展性、可靠性方面作一定的改进。如:在不同平台上与应用上对消息处理的通用性;当应用程序的逻辑关系需要扩展时,模型只需作出最少的改动,从而进行重用;客户端能够根据回调对象的状态进行响应处理,提高可靠性。因此,本文将利用代理的思想,设计一个具有代理节点的 AMI 通信模型。

1 已有的 Callback 模型

CORBA 中消息服务规范对 AMI 定义了两种通信模型,分别是 Callback 回调与 Polling 轮询[2],其中后者又是以前者为基础,因此本文以 Callback 为例,进行描述。

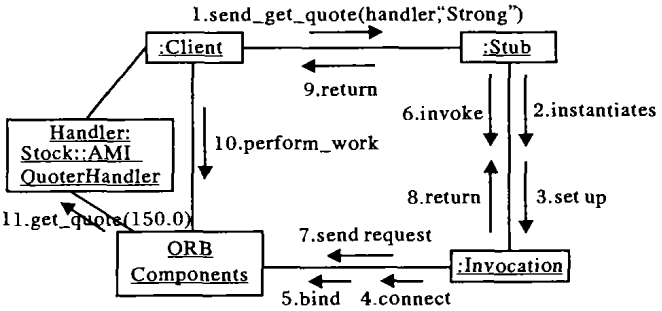

图 1 Callback 模型中的组件交互

Callback 模型结构如图 1 所示,模型实现的关键在于 reply handler servant 用 POA 的实现,因此,POA 的初始化与激

收稿日期:2004 - 07 - 20;修订日期:2004 - 10 - 12

作者简介:何智华(1979 -),男,广东清远人,硕士研究生,主要研究方向:软件工程、分布式消息通信; 邹北骥(1961 -),男,江西南昌人,教授,博士,博士生导师,主要研究方向:计算机图形学、图像处理、软件工程技术.

活是 Callback 模型的核心。

模型调度算法如下:

1) 调用发送含 AMI_QuoterHandler 对象引用和消息的异步操作。

2) 若 1)成功,则 Stub 实例化一个 Invocation 对象,代表本地的 ORB,与远端的 ORB 建立连接,并发送消息,同时把 AMI_QuoterHandler 对象引用存储在本地 ORB;若 1)失败,返回一个系统异常。

3) Invocation 对象返回控制权到 Stub,然后 Stub 把控制权返回给 Client。

4) Server 返回结果到 Client 的 ORB,ORB 调用在 2)建立的 AMI_QuoterHandler 对象,让该对象的方法分发给相应的异步调用方法,由此回调过完成。

2 Proxy 设计模式

一般的通信方式,是原点对目标点的直接通信。

可令 $X\{x_1,x_2,\cdots,x_n\}$ 为原点集,$Y\{y_1,y_2,\cdots,y_m\}$ 为目标点集,若每一个 y_m 都要面向每个 x_n,在其内部建立一套访问机制与服务机制,则 X 集与 Y 集之间总共的通信方式至多有 mn 种,而 Y 集的总开销为 mn,在此情况下,X 集中的各个元素,无法相互通信。

Proxy 是一种从出发点到目的地之间建立一道中间层,是一种变种较多的设计模式,其目的是为了消除原点与目标点的耦合性,并为第三方的访问提供支持[5]。思路如下:

同令 $X\{x_1,x_2,\cdots,x_n\}$ 为原点集,$Y\{y_1,y_2,\cdots,y_m\}$ 为目标点集,$A\{a_1,a_2,\cdots,a_n\}$ 和 $B\{b_1,b_2,\cdots,b_m\}$ 分别为 X 和 Y 的代理集,这时 X 集与 Y 集的通信,是通过在 A,B 代理集建立访问机制进行的,这时 X 与 Y 分别只需关心与各自的 A 和 B 建立影射关系,因此,X 集与 Y 集的通信方式至多有 C_{n+m}^2 种,是上述的 C_{n+m}^2/mn 倍。除了 X 集与 Y 集的元素进行通信之外,X 集内部和 Y 集内部的元素,可以进行两两通信。

以下情况将可用到 Proxy 思想:

1) 授权机制。当不同级别的对象集合元素对另外的同一对象集合元素拥有不同的访问权利,就必须通过代理集合元素,来对其访问进行控制。

2) 间接互动。对象 A 不能直接操作某个对象 B,但又必须和对象 B 有所互动。如对于开销比较大的对象,只有在使用它时才创建,这样可以节省许多宝贵的系统开销。

3 优化方法实现

3.1 方法设想

由以上对 Callback 模型的分析可知,Client 是通过 ORB 同时发送字符串消息和一个 AMI_QuoterHandler 对象引用到 Server,然后返回。在 Server 进行同步处理后,再返回 AMI_QuoterHandler 到 ORB,然后再调用相应的方法,来获取异步的,因此是"客户端的异步","C"和"S"的耦合还是比较强,而且客户端是被动接收回调通知,当服务器出现故障,客户端无法判断回调对象是否已经丢失。

因此,本文设想引入 Proxy 的思想,令客户端线程集合为 $X\{x_1,x_2,\cdots,x_n\}$,服务器端线程集合为 $Y\{y_1,y_2,\cdots,y_n\}$,两者之间添加一个"AMI_QuoterHandler 服务器",设其线程集合 $Z\{z_1,z_2,\cdots,z_n\}$,使得客户端与服务器端能通过"AMI_QuoterHandler 服务器"进行交互,对模型中所有的 AMI_QuoterHandler 对象引用进行排队、调度、管理,并开放对外接口,可把字符串的消息,转化为 XML 格式[6]的消息,实现对消息属性、服务质量和消息体的扩展,从而解决以上的问题。

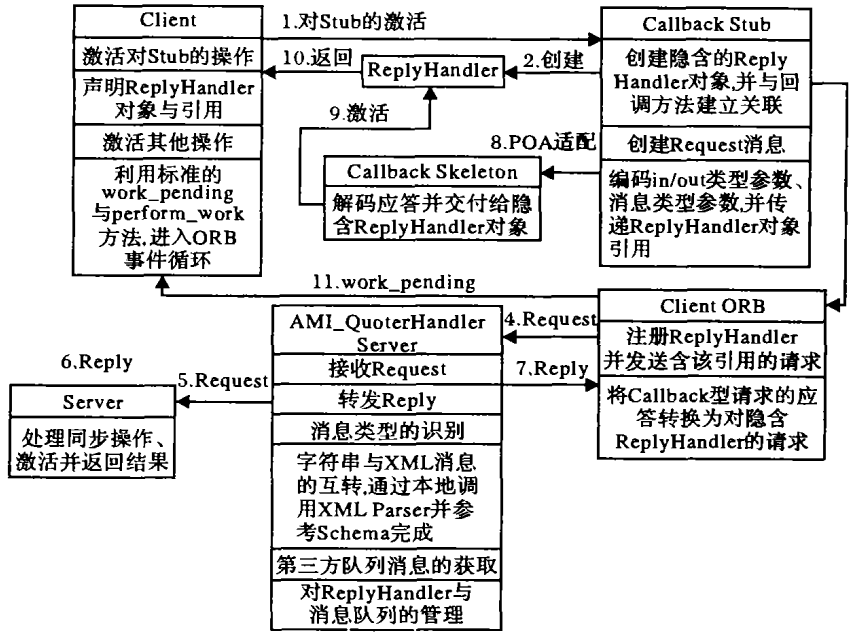

图 2　增加了 AMI_QuoterHandler 服务器的 Callback 模型

3.2 模型结构

本文提出的增加了"AMI_QuoterHandler 服务器"的 Callback 模型如图 2 所示。

3.3 核心组件

由于本模型是对 Callback 模型的改进,因此这里着重介绍与 Callback 相区别的核心组件,其主要功能如下:

1) Callback Stub:增添了一个 saveObject 的方法,使其能以简单的数据库或文本方式持久存储回调对象,从而不会使回调对象消失,保证了注册信息一直有效,直到回调信息返回,相应的注销操作执行。

2) Client ORB:增加了对 AMI_QuoteHandler Server 发送消息的接口,可根据 Client 的需要,将消息直接发送或者通过 XML Parser 转化 XML 格式再发送。

3) XML Parser:遵循本地 Schema 定义,对字符串和 XML 的 DOM 进行双向解析,考虑到通用性,该组件封装在 AMI_QuoteHandler Server。

4) AMI_QuoteHandler Server:封装了的一个组件,对外包含了消息接口,用于对消息的操作;对内进行消息的排队,调度等管理,并设置了相应的访问机制。

5) Server ORB:对从 Client 方发送的 AMI_QuoteHandler 对象引用和消息进行接收,经内部同步处理完毕后返回。

3.4 实现接口与核心源代码

接口的实现,要求在原有的基础上,通信双方实现对 AMI_QuoteHandler Server 的接口支持;而在 AMI_QuoteHandler Server 上,设置收发 XML 消息的接口、消息调度接口,可以根

据需要,实现对字符串或者是 XML 消息的输入与输出,以及对消息的排队管理。

1)定义 IDL 和 Callback Stub

通过定义以下的 IDL 接口,来证明是如何实现本优化模型的:

```
module Stock
{
 interface Quoter{
 //取得当前存储的值
 long getValue( in string stockname);
 };
};
```

在 Stub 中,将会在发送消息的方法中,添加消息标记的参数"messagetype"。

2)定义 ReplyHandler 与 POA(Client Skeleton)

ReplyHandler 由支持 AMI 的 IDL 编译器自动生成,分发从服务器返回的消息到调用它的 Stub,然后进行回调处理。

对 POA 的定义,必须实现以上的 ReplyHandler,这里采用"一对多"的策略,对每一个 AMI 调用,实例化不同的 servant 来分辨不同的回调:

```
class Async_Stock_Handler: public POA_Stock::
  AMI_QuoterHandler
{
 public:
   Async_Stock_Handler( const char * stockname)
   : stockname_( CORBA::string_dup (stockname))
   {}
   virtual void getValue( CORBA::Long returnvalue)
   {}
 private:
   CORBA::String_var stockname_;
};
```

3) 定义 XML 消息结构

XML 消息由 3 部分组成:消息属性、消息内容和消息服务质量,结构定义如下:

```
Module serverIdl
{
 typedef string XmlType;
 typedef string ContentType;
 struct MessageUnit{
   XmlType xmlKey;                    //消息属性
   ContentType content;               //消息内容
   XmlType qos;                       //消息服务质量
 }
};
```

4) 消息服务接口

消息服务主要分为以下七组:消息的发送接口、消息接收接口、消息服务的连接接口、消息服务的断开接口、获取消息数据接口、浏览消息类型库接口、以及消息人列出列控制接口。

3.5 组件交互与 ReplyHandler 对象执行动作

为了实现异步以及耦合性的消除,本模型把请求发送、消息队列和应答获取分为三个独立的步骤。Client 和 Callback Stub 共同负责请求的发送,AMI_QuoterHandler 消息的转换、管理和调度,应答的获取由客户端隐含的 ReplyHandler 对象实现。调度算法如下:

1) 客户对 Stub 激活,调用其中的方法对 ReplyHandler 对象激活,并由此与相应的回调方法建立关联。

2) Stub 把 ReplyHandler reference 和 Request 发送至 Client ORB,ORB 保留 reference 副本,然后发送含有引用的请求发送。

3) AMI _ QuoterHandler Server 接收请求,并根据 MessageType 保留消息的字符串类型或转化为 XML 类型,向 Server 发送;Server 进行同步处理,返回 Reply 到 AMI _ QuoterHandler,再把结果返回到 Client ORB。

4) Client ORB 通过注册的引用,利用 POA 适配来调用 Callback Skeleton,并由此解码应答并交付给隐含 ReplyHandler 对象,最后利用 ReplyHandler 对象,调用客户端相应的回调函数。模型工作流程完毕。

ReplyHandler 对象是回调的关键,对象的执行过程分为两部分:接收正常应答部分和接收异常应答部分。对于接收正常应答部分,动作如下:将返回值、out 参数和 in 参数保存;调用相应的回调函数。对于接收异常应答部分,Reply Handler 动作如下:将异常封装对象保存;若异常封装对象中的异常为用户自定义异常,则调用相应的方法进行处理。

4 实验结果与应用实例

根据以上所得的模型与设计结果,在局域网进行模拟实验,IDL 编译器采用 Visibroker for Cpp 和 Visibroker for Java。其中代理节点运行于 Linux 平台,利用 Java 进行开发;多个 Client 与 Server 分别运行于两个 Windows 平台,用 Cpp 进行开发。通信接口由 IDL 统一提供。

首先,对模型的性能测试作对比,方法是分别对两个模型进行 100 次测试。在 Client 中向代理节点发送消息体长度为 100 位的字符消息,并捕获当前系统时间(精确到毫秒),当节点捕获消息后,继而进行 XML 封装转发,而 Server 只作 Client 的消息处理。最后 Client 获得返回的消息,捕获当前系统时间,前后时间相减,即可得一次通信消耗的时间。按以上过程循环 100 次,所得数据如表 1。

表 1 模型性能测试数据对比

测试模型	最小时间/s	最大时间/s	平均时间/s
Callback 模型	0.33	1.20	0.52
基于 Proxy 的 Callback 模型	0.35	1.21	0.55

由表 1 可知,代理节点的引入,会给性能带来一定的损失,但是两者相差基本不是很大。

其次,对模型的可靠性进行测试,方法是分别建立 1、20、40、80、300 个 Client 经过节点,向同一个 Server 发送消息,Server 同步处理后返回,在 Client 编写一段测试的脚本,分别用于生成不同数量的请求,统计已经发送与捕获返回消息的数量,而 Server 同样编写一个统计接受数量的脚本。具体的测试数据如表 2。

(下转第 452 页)

智能答疑模块虽然能极大程度上帮助学生解答问题，由于知识库的局限还是不能排除学生无法获得所需答案的可能性，这时可借助人工答疑的方式帮助完成答疑。学生可以通过网上提问向系统提交新问题，这些新问题将由专门的教师负责解答或者在 BBS 发帖子，最终的问题记录保存到知识库中。整个答疑系统通过这几步骤有计划的帮助学生完成答疑。

2.2 实验分析

本文在制造业公共技术及服务平台系统（MIE-CTSPS）中实现了智能答疑系统。系统采用 B/S 结构。系统包括的功能有：学生注册、问题库管理、专业词库管理、智能答疑、人工答疑和网上交流。

实验分别采用常用的相关度算法和改进的算法对问题相关度进行测试，除了进行测试的 100 道题外，还请了几名学生对相关度算法进行评估，得到了比较好的结果。

计算相关度算法准确率的公式如下：

准确率 = 答对的问题总数/问题总数

实验结果如表 1 所示。

表 1 相关算法准确率

测试问题样本/个数	准确率		
	算法一	算法二	改进算法
60	0.467	0.573	0.809
100	0.339	0.394	0.770
150	0.203	0.279	0.731

很明显，系统采用改进后的问题相关度算法比常用的算法有了大幅度的提高。算法从词性及词频角度衡量权值，采用 BP 模型进行学习并建立学习模型计算相关度，实验证明具有实用性，提高了系统的准确率。

3 结语

针对现有网络教育平台答疑方式的不足，即人机交互方式不友好和缺乏智能性，开发了一个智能答疑系统。

1）学生可以采用自然语言方式对系统进行提问，提高了人机交互方式的友好性。

2）可以对自然语言形式的提问进行自动分词，这样就可以通过关键词集合的相关度来计算问题相关度。

3）利用有监督的机器学习，从关键词词性和词频角度划分特征向量，利用 BP 模型学习训练建立一个较好的学习模型优化分词权值，以提高问题相关度准确性。

但是需要指出的是，如果可以通过语义进行问题匹配会取得更好的结果，这也正是本文下一步研究的内容。

参考文献：

[1] 孙增圻，张再兴，邓志东. 智能控制理论与技术[M]. 北京：清华大学出版社，1997.

[2] 柳泉波，黄荣怀，何克抗. 智能答疑系统设计与实现[J]. 中国远程教育，2000.

[3] LITKOWSKI KC . Question Answering Using XML - Tagged Documents[J] , TREC-11 Text Retrieval Conference 2002.

[4] ELWORTHY D. Question Answering using a Large NLP System Proceeding of night Text Retrieval Conference[J]. Text Retrieval Conference, 2000: 156 - 165.

[5] BIN T, CHEN J, YI k. A Chinese Word dividing algorithm based on staticstical language models[J]. Proceedings of ICSP, 1996: 805 - 808.

（上接第 448 页）

表 2 模型可靠性测试数据（单位：个）

客户端访问数量	服务器接收数	客户端返回数	平均丢失数量
1	1	1	0
20	20	20	0
40	40	40	0
80	80	79	1
300	299	298	2

由表 2 可知，模型的误差率在小于 300 个访问线程时低于 1%，总的误差适中。基于 Proxy 的 Callback 模型比已有的 Callback 模型多了支持消息类型选择、应用环境无关、第三方订阅、回调对象确认等服务功能，这些功能，在具体的商业应用集成中是非常有用的。

本文的模型将应用在湖南省 AVIM 项目中的业务数据运营支撑系统，为其中的车驾基本档案管理系统、车辆监控系统、交通管理系统、门禁与停车收费系统、数据信息交易系统等，提供相应的通信支持。利用 XML 与 AMI 的 Callback 的结合，大大提高了平台无关性与通用性，为日后业务功能的扩展，提供了良好的通道。

5 结语

本文把 AMI 中的 Callback 模型与 Proxy 思想两者相结合，提出了一种具有代理节点的 Callback 模型，解决了模型本身技术不足，同时也扩展了 CORBA 中消息服务功能，使得通信更具有交互性。但是对耦合性的解除，同时也增加了系统一定的复杂性。在 CORBA 分布式系统中使用本地永久消息和 XML，是一个通用性与运行效率权衡的问题[7]，因此，在实际工程项目中应适当使用。

参考文献：

[1] formal/02 - 06 - 65, CORBA Components(Version 3.0)[S]. 2002 - 06.

[2] formal/04 - 03 - 20, CORBA Messaging chapter(Version 3.0.3)[S]. 2004 - 03.

[3] ARULANTHU AB, O' RYAN C, SCHMIDT DC, *et al*. The Design and Performance of a Scalable ORB Architecture for CORBA Asynchronous Messaging[R]. USA: Department of Computer Science Washington University, 2000. 208 - 228.

[4] 张小明，吴泉源，黎建良. 基于 CORBA 的异步调用技术[J]. 计算机应用研究，2001，(12)：28 - 31.

[5] 阎宏. Java 与设计模式[M]. 北京：电子工业出版社，2002.

[6] YERGEAU F, BRAY T, PAOLI J, *et al*. Extensible Markup Language (XML) 1.0 (Third Edition)[EB/OL]. http://www.w3.org/TR/2004/REC - xml - 20040204/, 2004 - 07.

[7] 王黎敏，陈奇，俞瑞钊. XML 与 CORBA 的结合：提高分布式系统的可扩展性[J]. 计算机科学，2002，29(5)：4 - 8.