计算机科学1999Vol.26№.8

CORBA 的异步消息处理

Asynchronous Message Processing in CORBA

骆志刚 唐雪飞 刘锦德

(电子科技大学计算机科学与工程学院 成都610054)

Abstract Processing asynchronous message is the limitation of current CORBA. Currently, the resolution is integrating Message-Oriented Middlware(MOM) with CORBA. At the same time, OMG is formulating CORBA messaging specification to overcome the limitation. The specification will include Asychronous Method Invocation(AMI), Time-Independent Invocation(TII)and general Quality of Service(QoS)policy.

Keywords Asynchronous message, Message-oriented, Middleware, AMI, TII, QoS

1. 引言

CORBA 是OMG 推出的一个重要规范，是当前面向对象分布式计算的一个典型代表。基于 CORBA 的分布式对象中间件已经得到广泛的使用，如 IONA 的 Orbix，Inprice 的 Visibroker 等等。CORBA 为分布在网络中各种异质实体上的实用对象提供了良好的基础设施，使之能够跨越异种平台相互透明动用资源和协同工作。图1给出了 CORBA 的体系结构。

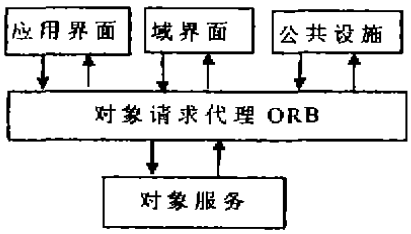

图1 CORBA 体系结构

ORB 是 CORBA 的核心，为客户和服务对象提供了透明合作的机制。它屏蔽了对象的位置，对象的实现细节，对象的执行状态和对象之间的通信机制。除了 ORB 外，CORBA 还提供了多项对象服务、公共设施、域界面和应用界面。应用开发者在这些基础之上开发分布式应用时，可以不必考虑底层的分布式编程问题，而将主要的精力放在上层的应用上。CORBA 将分布式计算推向了一个新的阶段，但它还有诸多的问题需要解决，其中异步消息通信就是当前 CORBA 的未曾解决的问题。

2. CORBA 的通信方式及其局限性

当前 CORBA 的调用模式基本上是同步请求/应答模式，在这一模式中，客户发出一调用请求后，等待服务器的应答，在客户的等待过程中，其线程处于阻塞状态，只有当它接收到服务器的应答后，才继续往下执行。在多线程的应用中，可以采取一个调用对应一个线程的方法，解决由于调用的阻塞而造成的整个应用的阻塞问题。但是多线程的使用会造成应用编程的复杂性并削弱了系统的可靠性。当前的 CORBA 中提供了三种方式试图解决同步请求/应答模式带来的问题：

1)单向调用(oneway)，采用单向调用，客户只发送调用消息，不等待服务器的应答(服务器也不作应答)，因而客户不会阻塞。这种方式缺少应答，一般是不可靠的。同时，在 ORB 与 ORB 之间的对象合作采用 IIOP 协议，一般情况下，IIOP 在 TCP 上实现，而在 TCP 这一层上，当与客户合作的服务器的 TCP 缓冲满时，它会将客户挂起，这样 IIOP 上的单向调用不能保证非阻塞模式。

2)延迟同步。这是一种同步请求/应答模式的变种，在客户发出请求后，不去等待应答，而是继续往下执行。过一段时间后，客户或查看服务器是否应答或执行一单独调用等待应答，这种模式能够达到异步调用的效果，但是它只能在动态调用(DII)中使用。动态调用的编程很复杂，首先要进行一系列的操作创建请求，之后才会发送请求消息。对于通常使用的静态调用(SII)无法使用这种模式。

3)事件服务。事件服务可以完成真正的异步消息传递。它定义了两个角色:事件提供者和事件消费者,通过事件通道完成它们之间的消息传递。但是事件服务提供的功能很简单,主要用于完成通知操作。

对于用户来说,需要更好的通信方式来保证消息的可靠性、消息的持久性和消息的服务质量等要求。在当前 CORBA 没有解决这一问题的规范的情况下,各厂商主要采用的一种简单方式是将已有的面向消息的中间件(MOM)与 CORBA 相结合、利用 MOM 在消息处理方面的完备功能,克服 CORBA 的这一缺点。同时,OMG 也在积极制订这方面的规范,以使将来的 CORBA 有自己的异步消息处理功能。

3. MOM 与 CORBA 的结合

3.1 MOM 的一般模型

MOM 通常使用消息队列完成进程(同一平台或不同平台)间的通信,其模型如图2所示,操作步骤如下:

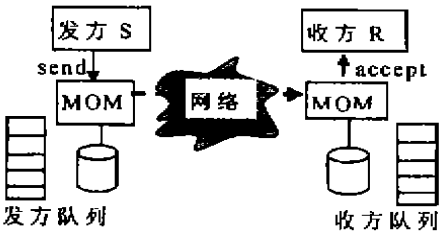

图2 MOM 通信模型

1)发方 S 调用 send 原语提交消息 M,消息提交后,发方不再关心任何消息发送的细节。

2)本地 MOM 收到提交消息 M 后,一是存储消息,二是向远端 MOM(可能在本地))发送消息 M。若远端 MOM 不可用,MOM 会周期性地发送直到远端可用或消息过期。

3)远端 MOM 收到消息 M 后,一是存储消息,二是将消息发送给收方 R。如收方 R 不可用,则远端 MOM 会周期性地发送直到收方 R 可用或消息过期。

4)收方 R 调用 accept 原语接收消息 M。

MOM 在消息处理上的技术十分成熟,它能完成异步传送,单点或多点传送,消息持久性(即存储转发)等功能,以及能满足各种服务质量(QoS)的要求。

3.2 CORBA 与 MOM 的结合模型

利用 MOM 在消息处理方面的优势,正好可以弥补当前 CORBA 在这方面的不足。图3给出了两者结合的模型。从图3中可以看出,为了使 CORBA 能与 MOM 合作,在 CORBA 中增加了一个消息适配器 MA(Message Adapter)。在 CORBA+MOM 系统中,普通的 CORBA 对象(即不使用 MOM 传递消息)与 CORBA+MOM 对象(即使用 MOM 传递消息)共存。对于 CORBA+MOM 对象,MA 负责完成它到消息队列的映射,CORBA+MOM 对象发送消息时,首先通过 MA 对 MOM 进行初始化,然后将消息传递给 MOM,以后的有关消息处理的任务就由 MOM 来完成。

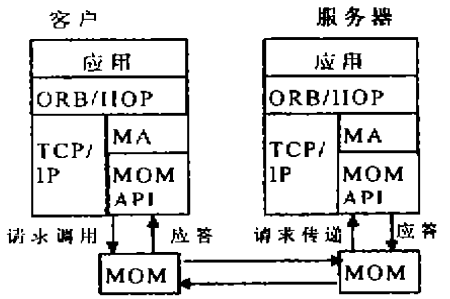

图3 CORBA+MOM 结构

这种 CORBA 与 MOM 结合的方式,集中了两者的优点。MOM 在消息处理方面给 CORBA 带来了以下优点:

1)使 CORBA 具有了完备的异步消息处理能力,应用对象可以做到真正的异步请求。

2)支持消息的持久性。消息一旦发出,即使当时服务器或网络不可用的情况下,可以保证消息到达服务器一次且只有一次。

3)服务器可以支持多种接收消息策略,如先进先出(FIFO),按优先级等。

另一方面,将 CORBA 作为 MOM 的上层,可以给 MOM 带来以下的好处:

1)为 MOM 提供标准的对象界面。通过 CORBA IDL 和 CORBA 的对象基础设施,开发 MOM 的应用更加方便,解决 MOM 开发应用困难的问题。

2)为描述消息提供了很好的方式。通过 IDL 的支持,消息内容的描述更加方便和容易。

3)提供了互操作的机制。MOM 建立在 IIOP 上,通过 IIOP 可以完成 MOM 间的互操作。

CORBA 与 MOM 结合的方式,是当前解决 CORBA 异步消息处理的主要方法,象 IONA 的 Orbix 与 IBM 的 MQseries 的结合,Expersoft 的 CORBAplus 与 Modulus InterAgent MOM 的结合就是采用的这种方式。

尽管 CORBA 与 MOM 结合的方式容易实现异

步消息处理，但是 MOM 没有规范，各种产品差别很大，造成每一种 CORBA＋MOM 互不兼容，失去了 CORBA 本身具有的良好的互操作性。OMG 在发出 RFP For Messaging 后，得到积极的应答，消息处理的规范初步形成。

4. 消息规范

消息服务规范将成为 CORBA3.0的一个重要组成部分。它主要包含三部分：异步消息调用、独立时间调用和服务质量。[1]

4.1 异步消息调用(AMI)

从前面的分析我们知道，现阶段的 CORBA 缺乏对异步方法调用很好的支持，AMI 提供了两种调用方式：

回调(Callback)：在这种方式中，客户首先创建一个称为 ReplyHandler 的对象，并将 ReplyHandler 的对象引用作为调用消息的一部分发送给服务器。消息发送后，客户将继续往下执行而不等待服务器的应答。服务器完成操作后，通过 ReplyHandler 的对象引用将结果发送给 ReplyHandler 对象，由 ReplyHandler 完成异步消息的接收。在 TII 中，对应的 ReplyHandler 是一个具有持久生命周期的对象 PersistentReplyHandler。

轮询(Polling)：这种方式中，使用了一个 Poller 的概念。一个 Poller 是一个值(在 Object-by-value 规范中定义)。在轮询模式中，一个异步方法调用将返回一个 Poller，客户可以通过 Poller 获得来自服务器的应答状态。若服务器尚未应答，客户可以选择阻塞操作来等待应答的到来。在 TII 中的轮询模式中，Poller 对应一个具有持久属性的 PersistentPoller。对于轮询模式操作的步骤如图4所示：

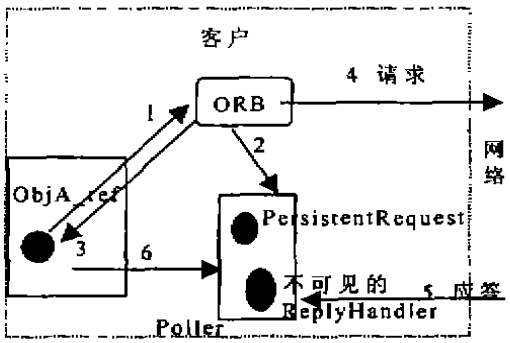

图4 轮询步骤

1)客户发送调用请求消息。

2)ORB 创建 PersistentReqnest 对象(用于 TII 中)和一个不可见的 ReplyHandler 对象，并将它们放入一个指定的 Poller 值中。

3)ORB 向客户返回 Poller。

4)ORB 将请求发送到网络中去。

5)服务器完成操作后，通过不可见的 ReplyHandler 的对象引用将结果发送给 ReplyHandler 对象。

6)客户通过 Poller 值请求应答，Poller 从不可见的 ReplyHandler 获得应答，并返回该客户。

AMI 无需多线程的支持就可以同时管理多个双向(request/reply)调用，使用回调或轮询解决了延迟同步或单向调用所带来的问题。

4.2 独立时间调用(TII)

TII 是一类在指定消息服务质量(QoS)下的特殊 AMI，其主要目的是支持"存储-转发"语义，即消息的持久性。在 TII 模式下，客户和服务器在不同的时间运行时，可以保证客户发出的请求能在未过期前传到服务器一次且只有一次，规范定义了一个标准的路由协议支持下面的方案：

·服务器可以是未激活的或不可激活的，客户发出的请求在 QoS 指定的生命周期内，使用"存储-转发"的机制到达目标对象。

·接收应答的客户可以是未激活的或不可激活的，服务器发出的应答在 QoS 指定的生命周期内，使用"存储-转发"的机制到达客户对象，接收应答的目标对象可以不是发出请求的目标对象。

作为 AMI 的特殊形式 TII，对 AMI 中的 ReplyHandler 和 Poller 中增加了持久性的属性，使之称为 PersistentReplyHandler 和 PersistentPoller，它们可以由其它进程实现，而不是发出请求的进程。同时还增加了：(1)PersistentReqnest，用来封装未完成的请求的对象，以支持延迟同步操作；(2)Router，软件路由器，用于目标对象(请求的目标或应答的目标)不可用时，消息的"存储-转发"；(3)IRP(Interoperable Ronting Protocol)，可互操作路由协议，TII 不是基于 IIOP 的，而是基于 GIOP。通过 IRP 可以支持比 IIOP 更高的 QoS 及与当前不同的 MOM 之间的互操作。

TII 的优点在于与时间无关，消息传递极其可靠，图5给出了一个操作结构示意。图5中表示客户向服务器发送异步调用请求，客户和服务器连接在各自的网络上。操作请求消息要经过 NetworkA 和 NetworkB 到达服务器，在这一过程中，可能出现 NetworkA、NetworkB、服务器或目标对象 ObjA 不可用，无论哪一种情况，消息不会丢失，一旦路径的下一部分可用，则消息会继续向前转发直到 ObjA 收到。图5中的路由器完成消息的存储-转发，如图5