计算机科学2001Vol.28№.4

CORBA 通讯机制分析

Analysis of the Communication Mechanism of CORBA

李天宁 周 刚 谢 立

(南京大学计算机科学与技术系 南京大学软件工程中心 南京210093)

Abstract With the development of CORBA structure and the updating of requirement of application, there is a trend of diversification in the communication mechanism of CORBA, which includes synchronous model, AMI, TII and so on. In this paper we present the comments on these models, then analyze the specification of quality of service about CORBA messaging.

Keywords CORBA, Synchronous, AMI, TII, QoS

一、概述

作为重要的分布式系统模型,OMG(Object Management Group)制定的CORBA(Common Object Request Broker Architecture)具有其他同类模型所不具备的优势:独立于平台、独立于语言、独立于操作系统,而基于CORBA的领域应用越来越多。随着现实应用需求的不断变化,对分布式系统下通讯方式的要求也日益多样化,这也带动了CORBA在这一方面的不断完善。本文就此分别介绍了CORBA规范[2]中提出的同步通讯模型、异步通讯模型和独立于时间的调用的工作方式,并对各自的优缺点及适用性作了较为详尽的分析;此外又对涉及消息处理的服务质量规范作了扼要的介绍和评述。

二、同步通讯模型及其变种

同步机制作为最基本的通讯模型,也是我们通常默认的服务调用方式。为了提高系统的并行度,OMG在早期的CORBA规范中又加入了one-way和延迟同步来模拟异步消息处理。

1. 同步方式(Synchronous):也称为双向同步。如图1所示,它要求Client向Server发出请求时,一定要得到回应之后,Client才能执行下去,而此前只能处于阻塞的状态。这种方式完全等同于本地方法调用的工作模式,程序书写十分直观,基本上符合人们的通常思维模式,但由于Client在未收到Server的响应之前完全处于阻塞状态,不仅占用了Client所在端的许多资源,也不能有效地响应异常,这样会极大地影响系统的灵活性。有鉴于这一原因,OMG提出了one-way和延迟同步两种模型用于改善系统的通讯性能。它们的思想是利用同步机制来模拟异步/部分异步通讯模型。

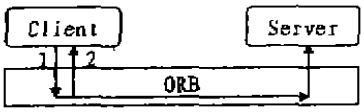
1. Client向Server发出请求并阻塞
2. Client得到Server响应后返回

图1 同步方式工作规程

2. One-way方式:通过分析现有方法调用,我们可以发现有一类方法的调用只对Server产生一定的影响,而对Client端却没有,因此OMG提出了one-way模型,也称为fire-and-forget模型。如图2所示,Client端发出方法调用后,无需等待Server的响应便可返回,这与COSS中的Notification服务很相似。很明显,这种方式下CORBA系统的效率很高。不过它对方法的要求也十分严格:方法的参数中不能有out/inout类型的变量,且方法也不能有返回值,因此使用的场合十分有限。第二,它并非真正意义上的非阻塞传输,而是对具体传输实体具有一定的依赖性:当一个one-way调用利用IIOP(Internet Inter-ORB Protocol)发送且运输层中的TCP缓冲区满时,one-way方式不能立刻从底层通讯协议栈中返回,即不能保证完全意义下的非阻塞,这必然会影响到系统的处理能力;而且在DCE Common IOR中缺乏对非同步语义的支持,因此one-way并非协议无关。第三,one-way采用了"尽力传送"语义,这也就意味着ORB只是在尽可能的情况下

李天宁 硕士研究生,主要从事分布式系统、计算机安全方面的研究。周 刚 硕士研究生,研究方向为分布式系统及并行计算。谢 立 教授、博士生导师,主要研究方向为分布式计算、并行处理、先进操作系统。

传送数据，而非完全保证传输过程的可靠性，因此它不适用于端到端的可靠传输。



1. Client 向 Server 发出请求
2. Client 不等 Server 响应而直接返回

图2 One-way 方式工作规程

3. 延迟同步(Deferred Synchronous)：在延迟同步方式下，Client 发出请求后可以继续下面的操作而不处于等待状态；为了得到请求返回的结构，Client 可以采用如下两种方式之一：(1)不断对 Server 轮询以获得响应；(2)由另外一个进程/线程代替自身等待返回。延迟同步方式一定程度上减少了 Client 的等待时间，而且较 one-way 方式应用更为广泛。但是它也有许多不足之处。首先，延迟同步方式目前只能用 DII(Dynamic Invocation Interface)实现，因此程序编制十分繁琐，而且缺乏安全有效的类型检测机制，更重要的是导致系统的运行效率大打折扣。第二，无论是采用轮询机制还是分线程/进程，都会导致 Server 端过于繁忙、网络有效带宽急剧下降，而 Client 端的负荷也会增加不少。因此，延迟同步方式也不能在真正意义上应付异步请求的需要。

1. Client 向 Server 发出请求
2. Client 不等 Server 响应而直接返回
3. Client 在随后的某个时刻轮询获得结果

图3 延迟同步方式工作规程

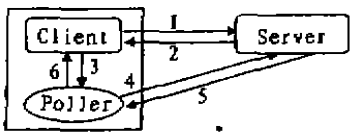
通过分析 one-way 和延迟同步方式，我们发现以同步方式模拟异步方式，将迫使我们使用进程/线程模型。这不仅导致程序变得复杂，而且就进程/线程模型本身而言，它并不能完全满足异步的需求，而且也会带来有关资源在各进程之间的同步问题。其次，如果我们使用双向 one-way 来模拟异步通讯(即 Client 采用 one-way 将请求发送到 Server 端，而 Server 端也利用 one-way 方式将结果返回给 Client)，将不可避免地存在如前文所说的 one-way 本身的缺陷。此外随着基于 CORBA 应用的不断推广以及 CORBA 的主要竞争对手 EJB(EnterpriseJavaBean)和 DCOM(Distributed COM)功能的不断完善，工业界与学术界都希望 OMG 能制定出真正意义上的异步消息处理规范。因此，OMG 在 CORBA 2.3及其后续版本中新增加了 AMI(Asynchronous Method Invocation)、TII(Time-Independent Invocation)和 QoS(Quality of Service)规范[5]以满足需求，其中 AMI 和 TII 作为真正的异步通讯模型，可以从根本上提高整个系统的并行度；而 QoS 规范涵盖了同步和异步调用中有关服务质量的需求，同时又为实时 CORBA(Real-Time CORBA)、容错 CORBA(Fault-Tolerant CORBA)等新型 CORBA 模型提供了良好的语义规范，为它们的推广铺平了道路。

三、异步方法调用 AMI

异步方法调用定义了两种通讯模型：轮询和回调机制。

1. **轮询**(polling)

如图4所示，Client 发出调用请求，并立刻返回一个 ValueType 类型的 Poller，随后 Client 通过 Poller 方法对 Server 监控，它可以选择使用阻塞或非阻塞的方法获取状态及返回结果。这样将对结果的远程查询操作本地化，大大节约了 Client 的等待时间，这一点也区别与延迟同步方式。当然这种通讯模型在目前看来有一定的缺陷，主要表现在如下三个方面：(1)由于 Poller 是 ValueType 类型[4]的变量，即需要 OBV(Object-By-Value)的支持，而目前 OMG 所制定的 OBV 规范不仅复杂，而且远未成为一个相对稳定的可操作规范，因此真正意义上的 Poller 模型只有在 OBV 规范趋于成熟才能进入实用阶段；(2)Client 在使用 Polling 模型通讯时，必须增加 Poller 对象的定义，也就意味着需要用户实现和维护更大的代码量；(3)Client 端发出操作请求后仍然要轮询，尽管它由远程方式改为本地方式，但并没有从根本上解决对象对响应结果的等待，这一点与同步调用方式是相同。



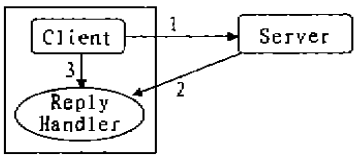
1. Client 向 Server 发出调用请求
2. Server 返回 Valuetype 类型的 Poller
3, 4. Client 通过调用 Poller 方法查询 Server 状态
5, 6. Client 取得返回的结果

图4 轮询机制工作规程

2. **回调机制**(callback)

如图5所示，Client 在调用 Server 的服务时，将位于本地的 ReplyHandler Servant 的引用传送到 Server 端，然后无阻塞地执行下去。当 Server 完成服务时，向 ReplyHandler 发出 Response 消息；然后 Client 便可以通过 ReplyHandler 获得服务返回的结果。回调机制方

式避免了由于轮询而给 Clent 端带来的负担；其二，它通过多个对象之间的协调回避了线程模型的引入，减轻了用户开发的负担；第三，我们从图中发现回调机制只对 Clent 端的程序设计有要求，而 Server 端的变化极小，这也利于提高系统部署的灵活性和可扩展性。但是实现回调机制的代码量较 Polling 方式更大，它不仅要改变 Clnet 的工作方式，更要实现 ReplyHandler 及其 Servant，这其中又包括正常情况下的接口和处理异常的接口。



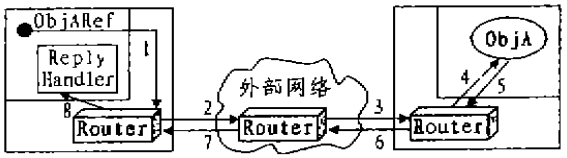
1. Client 将调用请求以及 ReplyHandlerServant 的引用传至 Server
2. 完成服务后，Server 调用 ReplyHandler
3. Client 从 ReplyHandler 获得 Server 间接的响应

图5　回调机制的工作规程

上述这两种通讯模型的优点在于：Client 端无需额外进程/线程完成异步调用，因此可以在单线程内提交多个异步请求，极大地节约了系统资源和等待时间；此外它也避免了由 one-way 和 DII 实现异步通讯带来的种种问题，提高了系统的性能。不过异步通讯模型由于其独特的工作方式较同步模型还是复杂了不少；其二，虽然 AMI 以 SII(Static Invocation Interface)方式实现，较 DII 方式的延迟同步方式简洁，具体实现过程中代码量依然不少，这又要求开发人员对其工作方式有比较深入的了解；另外，它还不能有效地处理各种异常。这些都是日后需要改进的地方。

四、独立于时间的调用 TII

我们可以把 TII 看作是一个特殊的 AMI，但与 AMI 不同的是，TII 中发出调用请求的对象与接收请求结果的对象可以不同，因此 TII 请求的生命周期可能与发出请求的对象的生命周期不一致。TII 的主要思想是对请求/响应的"存储转发"(Store-and-forward)，这很类似于电子邮件的工作原理。如图6所示，当调用请求所需经过的网络环境不好或 Server 暂时不可到达时，它先暂存于所经过的链路上的节点中，在网络条件允许的情况下，可以选择回调或 Polling 方式并利用 IRP(Interoperable Routing Protocol)为它们提供路由转发服务直至到达 Server 端对象，对返回结果亦然，因此 TII 这种工作方式特别适用于网络条件不好或移动用户。



1, 2, 3, 4. Client 将服务对象 A 的引用经过本地 Router、外部网络 Router 和 Server 端 Router 传至对象 A
5, 6, 7, 8. 对象 A 将返回结果利用传输链路上的 Router 反向传输至 Client 端的 ReplyHandler

图6　TII 工作规程

五、消息处理的服务质量

随着分布式应用的日益普及，人们越来越关注所得到的服务质量 QoS，它主要表现在端到端的响应延迟、请求优先级、传输可靠性等几个方面。而现有的 CORBA 规范中恰恰缺乏与基于消息处理机制(如 MOM，Message-Oriented Middlewear)系统相联系的 QoS。所以 OMG 在 CORBA 新规范中加入了有关 QoS 的规范说明[2]。它主要包括服务质量框架和服务质量策略等几个方面。

1. 消息处理的 QoS 框架

QoS 框架是 QoS 规范中最重要的组成部分。它较为完整地刻画了具有一定 QoS 特性的系统应遵循的模式和策略。QoS 框架分为 Client 端策略和 Server 端策略。

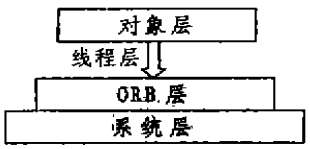


图7　Clinet 端 QoS 框架层次

Cleint 端 QoS 框架如图7所示。(1)系统层：作为 QoS 框架的缺省策略，它涉及十分广泛，但其中很大一部分并未标准化、实例化，所以在具体实施时，需要对它们进行重载或重定义，使之满足当前应用的需求。(2)ORB 层：它对系统层进行重载，用户可以通过策略管理器(Policy Manager)完成重定义。具体应用对 QoS 的要求不同会导致不同的 ORB 定义，因此在新规范下可能会出现一个应用需要不同的 ORB 支持的情况。(3)线程层：当 QoS 以线程为基本服务单位时，我们需要重载下面两层 QoS 策略，具体实现可以通过 Policy-Current 接口完成。(4)对象层：在新标准中最细的 QoS 策略粒度是对象，我们可以重载其下三层 QoS，并对每一个对象引用设立一个 QoS 策略。

在 Server 端策略中，QoS 策略与 Server 对象所在

的 POA(Portable Object Adapter)相一致,且在 POA 创建时设定。此外 Client 端的请求应遵循 Server 的 QoS 策略,以保证 QoS 的有效性。

2. QoS 策略

它主要包括如下几个方面:

(1)再绑定策略。传统意义下,我们对于因网络故障导致的路由失败是无能为力的,而在新规范中允许应用程序在透明绑定、连接关闭绑定和放弃绑定之间做出选择,使链路管理标准化。

(2)同步策略。在 one-way 方式下,请求发出后 Client 即可返回。而在何种情况下可以返回,如图8所示,OMG 定义了四个层次的同步策略(SyncScope)。(1)SYNC_NONE:在 Client 向 ORB 发出请求后便可返回,它的等待时间最短,适用于底层处理过长或具有一定容错性的系统。(2)SYNC_WITH_TRANSPORT:当 ORB 将请求压入 TCP 栈时 Client 便可返回,而无须 ACK 响应。(3)SYNC_WITH_SERVER:当请求送到 Server 端的 Servant 时,系统返回一个 ACK 后,Client 才可以返回,这种方式较为适合于实时系统;(4)SYNC_WITH_TRAGET:这种方式下,只有当服务完成,Client 才可以返回,即等同于一般意义下的同步响应,所以也是最为保险的同步机制。

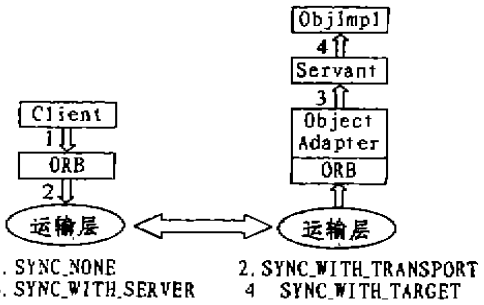


图8 QoS 的四种同步策略

(3)请求/响应优先级及其队列排序策略:在 TII 中由于有 Router 的存在,我们需要在转发请求/响应之前确定其转发的先后次序,而优先级的获得可以根据设置的不同分别由 Server 和 Client 确定。当一个请求到达某个 Router 时,应用程序有权决定进入等待队列的请求排入队列的方式,诸如时间顺序、优先级顺序、生命周期顺序等。

(4)请求/响应超时:请求/响应报文传输的时间包括本方系统处理时间、底层通讯时间和对方系统处理时间,对于等待时间敏感的系统(如实时 CORBA)就要有一定的约束机制以防止过长时间的等待。请求/响应超时机制保证了 Client/Server 在发出报文的一段时间后有一个确定的动作结果:正常返回或因超时异常而退出,而非不知期限地等待。

(5)请求路由策略:随着 AMI 和 TII 的出现,应用程序被赋予了自行决定请求路由的权力:既允许直接的发送,也允许有所选择地决定可经过的 Router 及其最多的个数等。

总之,QoS 策略为灵活的控制应用程序提供了一套标准,但是并不是说在具体应用中一定要使用 QoS 策略,因为开发人员至少要考虑如下两点:(1)程序复杂性:设置 QoS 策略是一项十分繁琐的工作,不仅会增加代码量、更要求开发人员对应用需求十分明确;(2)可移植性:如果对应用的 QoS 设置过多,则会增加对它的限制,这当然也就限制了在不同运行环境之间的可移植性。

结束语 本文上面介绍了 OMG 提出的几种通讯模型以及与消息处理有关的 QoS,并分析了它们各自的优缺点。从中我们可以看到每一种技术都不是十全十美的,这不仅需要不断完善自身,更需要加强各规范/模型之间的联系。比如消息处理的 QoS 规范不只针对目前的同步、异步通讯模型,还涵盖了新近推出的实时 CORBA、容错 CORBA 和 Minimum CORBA 等新的 CORBA 模型。此外,在 CORBA 2.3及其后续版本中加入了双向 GIOP(bi-directional GIOP)[3]的规范,这为提高 CallBack 的性能开辟了新的途径。总之,CORBA 通讯机制及其相应设施的不断完善必将为基于 CORBA 应用的推广打下一个坚实的基础。

参考文献

1 Otte R, Patrick P, Roy M. Understanding CORBA. Prentice Hall PTR, 1996

2 Object Management Group. CORBA Messaging Specification. May 1998

3 Siegel J. A Preview of CORBA 3. Computer, 1999(May): 114~116

4 Vinoski S. New Features for CORBA 3.0. Communication of The ACM, 1998(Oct.) 44~52

5 Schmidt D C, Vinoski S. An Introduction to CORBA Messaging. SIGS C++ Report, November/December 1998