CORBA 异步消息的研究与实现

张志伟[1], 郭长国[1], 蔡俊亚[2], 吴泉源[1]
(1. 国防科技大学计算机学院网络技术与信息安全研究所, 湖南长沙 410073; 2. 湖南商学院信息管理系, 湖南长沙 410073)

摘 要: 异步机制是构造大规模分布式系统必不可少的机制之一. 作为一种典型的分布应用支撑平台, CORBA 没有很好的解决异步机制问题, 这限制了 CORBA 在一些领域的应用. 如何在 CORBA 中提供异步通信支持, 为上层应用提供异步通信支持成为研究的热点. 本文提出一种 CORBA 异步消息模型 Star2Async, 该模型通过 ReplyHandler 对象实现应答处理、通过异常封装对象实现异常处理, 通过基于修改抽象语法树的机制实现异步代码生成. 在自主研制的分布对象中间件平台 StarBus 中实现了本文提出的 Star2Async 异步模型, 实现和初步应用表明本文提出的异步消息模型为在 CORBA 中实现异步机制提供了一种有效的参考.

关键词: CORBA 异步消息; 异步机制; 回调模型; 轮询模型
中图分类号: TP3111 52 文献标识码: A 文章编号: 037222112 (2004) 1121820204

Research and Implementation of CORBA Asynchronous Messaging

ZHANG Zhi2wei[1], GUO Chang2guo[1], CAI Jun2ya[2], WU Quan2yuan[1]
(1. Institute of Network Technology & Information Security, School of Computer Science, National University of Defense Technology, Changsha, Hunan 410073, China;
2. School of Information Management, Commercial College of Hunan, Changsha, Hunan, 410073, China)

Abstract: Asynchronous mechanism is absolutely necessary in the construction of large scale distributed systems. Although being a popular support platform for distributed applications, CORBA doesn't solve the asynchronous communication mechanism problem very well and this limits its application to some application domains. How to provide asynchronous mechanism in CORBA and provide asynchronous communication support for applications at upper layer have become the research hotspot. This paper presents a CORBA asynchronous messaging model Star2Async. The model supports reply processing, exception handling and asynchronous code generation by means of ReplyHandler object, exception encapsulation object and modification of abstract syntax tree accordingly. We have implemented Star2Async model on StarBus, which is our own distributed object2oriented middleware platform implementation. The implementation results of Star2Async and its initial applications show that Star2Async model provides a reference model for the implementation of asynchronous mechanism in CORBA.

Key words: CORBA asynchronous messaging; asynchronous mechanism; callback model; polling model

1 引言

异步机制是构造大规模分布式应用必不可少的重要机制之一. 分布对象中间件作为一种典型的分布式应用支撑平台, 没有很好地解决异步机制问题. 缺乏对消息的异步传递、服务质量控制和存储转发等提供支持被认为是分布对象中间件的一大缺陷[1]. 在分布对象中间件中提供异步通信支持成为研究的热点, 目前相关的工作主要集中在两个方面: 首先, 通过中间代理服务解耦请求发送方和接收方, 实现异步、松耦合和多对多通信, 如 CORBA 中的事件服务和通告服务, J2EE 中的 JMS; 其次, 从方法级支持请求发送方和接收方之间的异步通讯, 如 CORBA 中的异步消息[2].

CORBA 作为一种主流的分布对象中间件没有很好的解决异步机制问题. 传统的 CORBA 只提供同步激活机制, 随着 CORBA 应用领域的扩展, 这一特性已经不能满足需求. 虽然通过 CORBA 中的 one2way 和 DII 机制可以取得一定的异步性, 但是它们都不是真正的异步机制, 其中 one2way 机制存在对传输层的依赖和 best2effort 语义问题, DII 机制存在编程复杂、过多的内存分配和数据拷贝等问题[3]. 虽然 CORBA 事件服务和通告服务从服务的层次对异步消息提供支持, 实现了异步、松耦合和单点对多点的通信模型以及事件过滤等功能, 但它们基本上只适合建立基于事件的中间件应用, 还存在不足之处, 如不支持消息消费者向生产者返回应答, 只支持有限的几种参数类型, 不能在方法级提供异步支持等.

正是因为上述原因, 如何在 CORBA 中提供异步通信支持, 为上层应用提供有力的异步通信支撑成为研究的热点, 许多组织都针对 CORBA 异步消息开展了研究, 比较重要的两个系统是 TAO 和 Orbix. TAO 目前推出了支持异步回调(Call2back)模型的产品[3]. 除了支持异步回调模型之外, TAO 最近提出了 AMH(Asynchronous Method Handling)机制, 该机制允许

收稿日期: 2003201220; 修回日期: 2004207220
基金项目: 国家自然科学基金(No. 90104020); 863(No. 2001AA113020, No. 2003AA115410)

服务方异步地处理客户请求[4]. Orbix 也推出了支持回调模型的系统.

本文基于国防科技大学的分布对象中间件平台 StarBus 对 CORBA 异步消息进行了研究[5~8], 提出了一种异步消息模型 Star2Async, 该模型通过应答处理对象(ReplyHandler)实现应答处理, 通过异常封装对象实现异常处理, 通过基于修改抽象语法树的机制实现异步代码生成. Star2Async 的突出贡献在于提出了一种基于隐含 ReplyHandler 的轮询(Polling)模型应答处理机制. 实现结果和初步应用表明本文提出的异步消息模型对于在 CORBA 中实现异步消息提供了一种有效的参考.

2　Star2Async 异步消息模型

2.1　系统结构

图 1 给出了本文提出的 Star2Async 异步消息模型. Star2Async 包括两个子模型, 即回调模型和轮询模型, 图中左边为异步回调模型, 右边为异步轮询模型. 这两种模型都需要 ORB 提供支持, 本文将除了提供常规 ORB 功能之外同时支持这两种异步模型的 ORB 称为异步 ORB(Messaging ORB).

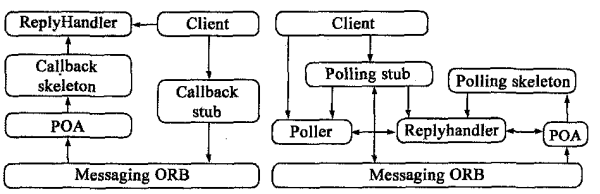

图 1　Star2Async 异步消息模型

下面讨论模型中主要组件的功能:

¹ ReplyHandler: 用于接收应答的 CORBA 对象. 回调模型中客户直接与 ReplyHandler 对象交互, 轮询模型中 ReplyHandler 对象对客户是透明的.

º 异步 ORB: 除了常规的 ORB 功能之外, 主要完成 ReplyHandler 对象的注册、识别异步请求的应答并将 GIOP Reply 消息转变为对 ReplyHandler 对象的 GIOP 请求.

» Poller 对象: 遵循对象传值标准的值类型对象, 轮询模型中的客户通过该对象查询应答状态和获取应答.

¼ Callback Stub: 注册客户提供的 ReplyHandler 对象并将请求信息编码, 将编码后的请求消息通过异步 ORB 发送出去.

½ Polling Stub: 创建 Poller 对象和 ReplyHandler 对象, 建立二者之间的关联, 将编码后的请求信息通过异步 ORB 发送出去.

2.2　Star2Async 回调模型

为了实现异步性, Star2Async 回调模型把请求发送和应答处理划分为两个不同的处理过程: 请求发送由客户和 Callback Stub 共同负责, 应答处理由 ReplyHandler 对象负责.

Callback Stub 是应用层和异步 ORB 内核之间的连接纽带, 为应用层提供了一个强类型的静态调用接口, 能够对应用层的请求参数进行类型安全检查, 并负责将这些参数编码为 GIOP Request 消息, 然后通过异步 ORB 发送出去. 因为构造的请求消息与普通的请求消息一样, 所以服务方不必区分到达的请求是普通请求还是异步请求. Callback Stub 也需要将 ReplyHandler 对象的引用传递给内核进行注册.

ReplyHandler 对象负责接收异步请求的应答, 完成应答处理功能. 客户需要实现该对象完成自己的应答处理逻辑. 当应答返回后, 异步 ORB 根据请求 ID 识别出是回调型请求对应的应答, 提取出应答结果并将它变换为针对注册的 ReplyHandler 对象的请求, 这个请求将通过正常的请求处理过程到达 ReplyHandler 对象.

回调模型组件之间的交互如图 2 所示.

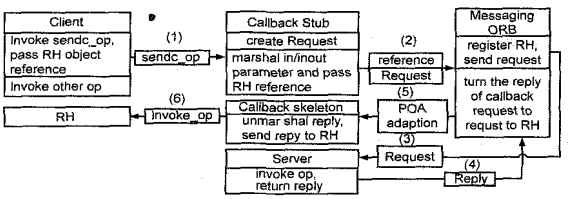

图 2　Star2Async 回调模型组件交互图

2.3　Star2Async 轮询模型

为了实现异步性, 轮询模型把请求发送、应答接收和应答获取分为三个独立的过程. 客户和 Polling Stub 共同负责请求发送, 应答接收由隐含的 ReplyHandler 对象实现, 应答获取由客户利用 Poller 对象完成.

Polling Stub 在模型中具有重要地位, 它负责创建隐含 ReplyHandler 和 Poller 对象并建立二者之间的关联. 客户利用 Polling Stub 发送请求, 之后返回一个类型相关的 Poller 对象, 利用该对象获取应答结果. 请求经过异步 ORB 发送到服务方, 同步地在服务方执行. 返回的结果首先到达客户方异步 ORB, 它识别出是一个轮询型请求对应的应答之后将该应答转换为针对隐含的 ReplyHandler 对象的请求. 隐含 ReplyHandler 负责保存应答, 供 Poller 对象查询.

轮询模型组件之间的交互如图 3 所示. 图中所示的是一种客户查询时应答还没有返回, 客户阻塞等待应答的情况. 实际的查询模式有三种, 即阻塞、非阻塞和有时间约束的阻塞.

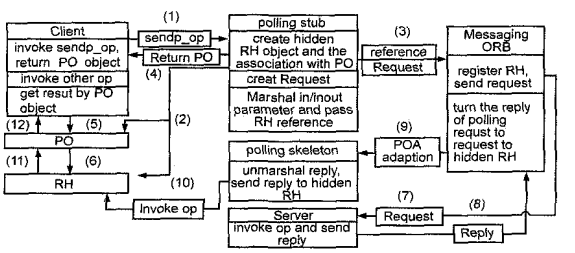

图 3　Star2Async 轮询模型组件交互图

3　核心机制

在传统的同步方法调用机制中, 客户发送完请求后阻塞等待应答. 但是在异步方法调用机制中, 客户发送完请求之后并不阻塞等待, 而是立刻返回. 必须解决由于异步性而产生的三个问题: 应答返回后如何接收、保存和传递; 如果方法引发了异常, 异常如何保存、传递和重新引发; 编译器如何生成支持异步机制所需的代码.

下面讨论与上述问题相关的三种核心机制: 基于隐含 ReplyHandler 的轮询模型应答处理机制、基于异常封装对象的异常处理机制和基于修改抽象语法树的代码生成机制.

3.1.1 基于隐含 ReplyHandler 的轮询模型应答处理机制

与回调模型的应答处理机制相比,轮询模型的应答处理机制比较复杂,它必须解决如下两个问题:首先,如何接收和保存异步应答;其次,当客户发送/sendp_0 型请求后会返回一个对应的 Poller 对象,客户随后将利用这个 Poller 对象查询应答,但是对于一个接口定义中的所有/sendp_0 型操作都会返回相同的 Poller 对象,这就存在 Poller 对象如何区分同一接口中不同操作返回的结果和 inout/out 类型参数返回值的问题.

针对上述两个问题,本文提出一种基于隐含 ReplyHandler 的机制.隐含 ReplyHandler 在功能上与回调模型中的 ReplyHandler 一样,但在回调模型中请求发送方直接与 ReplyHandler 交互,在轮询模型中,请求发送方不直接与 ReplyHandler 交互.下面通过例子说明这种机制.

对于原始接口定义中的每一个方法(图 4(a)所示的 say_hello)都会生成一个隐含的 ReplyHandler 对象(图 4(b)),该对象有两个方法:第一个方法的声明(say_hello)与 Poller 对象中对应的方法(say_hello)相比缺少一个时间约束参数,其他参数除了方向性(in/inout/out)之外是一样的.Polling 异步请求(sendp_say_hello)的正常应答返回后,异步 ORB 将激活该方法(say_hello),将传递的每一个参数值保存起来供 Poller 查询.第二个为处理异常的方法,当异常应答返回后异步 ORB 将调用该方法保存异常数据.

对于一个接口中的每一个/sendp_0 型方法,都会返回相同的 Poller 对象名字(图 4(c)).但是因为 Polling Stub 在发送每一个请求时创建的 Poller 对象(图 4(c))与隐含 ReplyHandler(图 4(b))对象是一一对应的,所以 Poller 可以区分返回的是哪一个操作的应答结果.

```
exception ex1{string exname1};
Interface Hello{
long say_hello(in long a1 out long b1 inout long c) raise(ex1);}
```

(a)原始接口定义

```
Interface AMI_HelloPolling_say_helloReplyHandler{
Void say_hello(in long ami_return_value1 int long b1 in long c);
void say_hello_excep(in AMI_HelloExceptionHolder excep);}
```

(b)隐含 ReplyHandler

```
valutype AMI_HelloPoller:Messaging::Poller{
void say_hello(in long timeout1 long ami_return_value1
        out long b1 out long c);}
```

(c)Poller 对象

```
valuetype
AMI_HelloExceptionHolder:Messaging::ExceptionHolder{
void raise_say_hello()raise(ex1);}
```

(d)异常封装对象

图 4 接口定义和生成对象的接口描述

3.1.2 基于异常封装对象的异常处理机制

在 Sta2Async 回调模型和轮询模型中,客户发送完请求就立刻返回,正常应答的结果将通过参数的形式传递给对应的 ReplyHandler 进行处理.但是如果是异常应答,则不能通过参数的形式传递给 ReplyHandler.此外,客户也无法按照常规的方式捕获异常.这就存在如何封装异常和引发异常的问题.

针对上述问题,采用了基于异常封装对象(ExceptionHolder)的机制.异常封装对象的主要功能是封装异常数据并提供重新引发异常的方法.对于接口定义文件中的每一个接口,都生成一个对应的异常封装对象.图 4(a)中定义的接口所对应的异常封装对象如图 4(d)所示.

异步 ORB 发现应答是一个异步请求关联的异常后会激活 ReplyHandler 中对应的方法(/_except0 型方法).在方法到达实现对象之前将有对应的代理对象(skeleton)解码异常数据流并封装为一个异常封装对象,然后传递给 ReplyHandler 实现对象.因为异常封装对象是一个值类型的对象,所以可以在 ReplyHandler 的进程空间中重建.对于回调模型,客户需要在 ReplyHandler 中编写代码,通过异常封装对象中的方法获取异常.对于轮询模型,隐含 ReplyHandler 将保存传递给它的异常封装对象.当客户利用 Poller 对象查询应答时,Poller 对象将查询与之关联的隐含 ReplyHandler 对象,如果发现有异常返回,则通过异常封装对象中的引发异常方法(图 4(d)中的 raise_say_hello)重新引发异常,客户需要在自己的代码中捕获异常.

3.1.3 基于修改抽象语法树的代码生成机制

Sta2Async 需要 IDL 编译器生成支持回调模型和轮询模型的代码.主要包括异步请求编码和请求发送代码、解码应答对象代码、查询应答对象代码、异常封装对象代码和接收应答对象代码.这样就存在一个如何定制 IDL 编译器,使之支持异步机制代码生成的问题.

传统的机制采用基于隐式 IDL 文件的异步代码生成,如图 5 所示.这种机制的主要思想是扫描原始接口定义文件,按照规则生成一个隐含的 IDL 文件,该文件中包含了支持异步所需的接口定义,然后再以该文件为输入进行前端语法分析,生成抽象语法树,后端进行代码生成.这种方法的主要缺点是需要进行两次预处理和语法分析,效率不高.

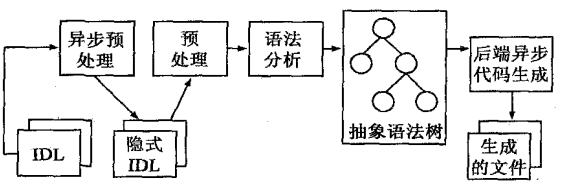

图 5 基于隐式 IDL 的异步代码生成机制

本文提出了一种基于修改抽象语法树的异步代码生成机制[5],如图 6 所示.与第一种机制相比,这种机制只需进行一次预处理和语法分析,效率较高.这种机制在抽象语法树生成

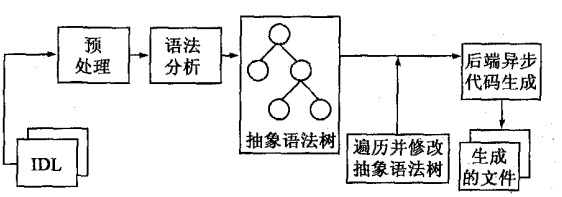

图 6 基于修改抽象语法树的异步代码生成机制

之后,代码生成之前直接修改抽象语法树,插入支持异步机制所需的节点.在后端代码生成时识别出这些异步节点,按照规则生成支持异步机制的代码.

4　测试

基于本文提出的 Star2Async 模型在 StarBus 中实现了异步消息子系统,该系统支持异步回调机制、异步轮询机制和一些服务质量控制策略.实现了支持异步代码生成的 IDL 编译器和异步 ORB,进行了异步性能测试.下面给出具体的异步性能测试结果.

411　Star2Async 异步测试

测试环境为国防科大并行与分布处理国家重点实验室内线路速率为 10Mb/s 的局域网,使用的对象中间件为国防科大自主研制的支持异步消息的分布对象中间件平台 StarBus.两台操作系统为 Windows 2000 Server 的 PC 机通过局域网连接,PC 机的配置为 1l 6GHz、256MB 内存.客户方构造请求并发送到服务方,以 10000 个调用为一个测试单位.

(1) 两种异步机制吞吐量测试

为了对 Star2Async 中两种异步机制的性能进行比较,测试了两种异步机制的吞吐量,如图 7 所示.图 7 说明在 Star2Async 中回调机制的吞吐量高于轮询机制的吞吐量.这符合实际情况,因为 Star2Async 轮询模型与回调模型相比,引入了更多的对象,增加了消息的执行路径的长度.

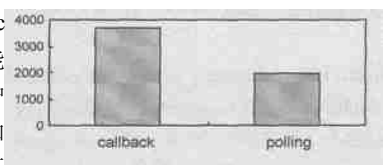

图 7　Star2Async 中两种异步机制吞吐量对比图

(2) Star2Async 轮询模型并发效果对比测试

图 8 给出了轮询模型的并发效果测试图.S2 对应的测试程序一次并发发送多个调用,返回多个 Poller 对象,然后利用每一个 Poller 对象获取对应的结果.S1 对应的测试程序一次发送一个调用,只返回一个 Poller 对象,然后利用该 Poller 对象获取结果.测试结果表明,利用轮询模型的并发特性后,平均延迟得到了改善.

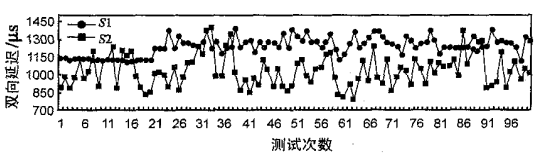

图 8　Star2Async 轮询模型并发效果测试对比图

412　与同类系统的对比测试

根据作者的调查,目前没有同类的支持轮询模型的产品,所以本文仅给出了与支持回调模型的 TAO 的对比测试结果.选择了与 ACE512+ TAOl12 环境中的回调模型进行对比测试.测试环境同 411,但是测试程序不同.图 9 说明两个系统中的回调机制性能相当,需要进一步对 Star2Async 中的轮询机制进行优化.

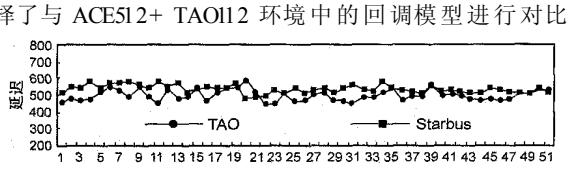

图 9　与 TAO 的对比测试

5　结束语

由于缺乏对异步消息提供支持,与其他的开放式分布处理标准相比,CORBA 在支持大规模分布式处理方面相形见绌.在 CORBA 中提供异步通信支持,为上层应用提供有力的异步通信支持对于将 CORBA 的应用领域扩展到大规模分布式系统具有重要意义.本文讨论了我们提出的 Star2Async 异步消息模型和其中的核心机制,给出了性能测试和与同类系统的对比测试.实现结果和初步应用表明,Star2Async 模型为在 CORBA 中实现异步消息提供了一种有效的参考.

进一步的研究工作包括基于异步消息研究对象中间件的消息存储转发机制,实现时间无关的激活语义,使 CORBA 能够支持断开的客户/服务器应用和移动应用.

参考文献:

[1] Steve Vinoski. New features for CORBA 3.0[J]. Communications of the ACM, 1998, 41(10): 44- 52.

[2] Object Management Group. The Common Object Request Broker: Architecture and Specification, 2.5 ed, 2001[S].

[3] Douglas C Schmidt. The design and performance of a scalable ORB architecture for CORBA asynchronous messaging[A]. IFIP/ACM International Conference on Distributed Systems Platform[C]. New York, USA, 2000. 208- 230.

[4] Mayur Deshpande, Douglas C Schmidt, Carlos O. Ryan, Darrell Brunsch. Design and performance of asynchronous method handling for CORBA[A]. Proceeding of the Distributed Objects and Applications (DOA) Conference[C]. Irvine, CA, 2002. 568- 586.

[5] 张志伟. 分布对象中间件中的异步机制代码生成研究[J]. 计算机工程与应用, 2003, 39(6): 41- 43.

[6] 王永恒. CORBA 异步消息的研究与实现[D]. 长沙: 国防科技大学研究生院, 2001.

[7] 郭长国. 基于 CORBA 消息服务的容错机制研究[J]. 计算机学报, 2002, 3(10): 1059- 1064.

[8] 刘步权, 王怀民, 姚益平. YH2SRTI 软件中的关键技术[J]. 软件学报, 2003, 14(2): 1148- 1155.

作者简介:

张志伟　男, 1975 年生于河南邓州,空军指挥学院博士,研究兴趣为分布计算技术,数据库技术和智能软件.

郭长国　男, 1973 年生于河南武陟,国防科大博士,兴趣为分布计算,实时系统.