维普资讯 http://www.cqvip.com

CORBA 技术前沿*)

The Frontline of CORBA

曹晓阳　刘锦德

(电子科技大学计算机科学与工程学院　成都 610054)

Abstract　CORBA 3 is enhanced by some important specifications which are integrating with Internet, QoS and component model. It brings several new features to us, such as ease-of-use and precise control to establish architecture. These additions will ensure that CORBA continuse to play an ever-increasing role in the computing world of the future. The topic shows what are the details.

Keywords　CORBA, QoS, CORBA component

在 CORBA 3 的规范中引入了许多新的技术，使 CORBA 的功能得到扩展，使 CORBA 更易使用。虽然目前尚没有符合 CORBA 3 规范的产品，但是我们从这些规范中已经可以看到 CORBA 最新技术的整体结构。

CORBA 3 的改进共包含下列三个方面：与 Internet 集成、服务质量控制(QoS)和 CORBA 组件结构，本文将对这三个方面详加阐述。

1. 与 Internet 集成

在 CORBA 3 的规范中，有两项使 CORBA 与 Internet 的继承更为紧密，它们是防火墙规范和可互操作的名称服务规范。

1.1　防火墙规范[1]

CORBA 3 的防火墙规范的目标是 ORB 能更好地与防火墙协同工作，使客户通过防火墙到服务器的通讯能更好地被控制。该规范对传输层和应用层的防火墙进行了定义，此外它还定义了双向的 GIOP 连接，该种连接对 CORBA 的回调和事件处理非常重要。防火墙规范的定义可以保证基于 CORBA 的应用可以安全地运行在 Internet 上。

传输层防火墙允许使用不同的应用层协议来访问受保护的资源，完全根据传输包中的地址信息来控制资源存取，通常它只在建立连接时执行存取控制检查。应用层防火墙则严格限制使用某种应用层协议，如 IIOP 或 HTTP，即使在相同的主机和端口上的连接，不同应用层协议的通讯在通过防火墙时可能产生不同的结果。

传输层防火墙运行在 TCP 协议上，使用地址转换机制来实现对内部地址的保护。在 CORBA 防火墙规范中，指定了 TCP 的 683 端口用于 IIOP，684 端口用于使用了 SSL 协议的 IIOP。管理员可以对防火墙进行配置，使它能过滤基于 IIOP 的 CORBA 信息交流。对于 CORBA 则不需要更多的修改。

SOCKS 协议是传输层的网络代理协议，在基于 TCP 或 UDP 的客户和服务器对象之间建立作为数据通道的代理，该代理对于客户和服务器是透明的。对于 SOCKS，IIOP 仅仅是一种基于 TCP 的应用层协议，它能为 IIOP 提供代理机制。CORBA 的客户程序不需要修改源代码、只需链接支持 SOCKS 的 TCP 库，并配置所在主机将 SOCKS 请求发往指定的代理服务器，就可以通过 SOCKS 代理建立到 CORBA 服务器的连接。

在规范中还定义了 GIOP 代理防火墙，它是一种针对 GIOP 的应用层防火墙。GIOP 代理防火墙基于 GIOP 包中的信息完成存取控制。例如 GIOP 代理可以否决对某个特定对象的访问，或否决对对象的某个操作。

在应用中，CORBA 对象常常需要对调用它们的客户进行回调或触发事件，响应客户的调用请求。这种响应与客户调用对象的信息传递方向相反。但是标准的 CORBA 连接只能进行单向的调用请求，回调则需要建立第二个反向的 TCP 连接，这样对于防火墙是不能接受的，新建的连接通常是无法通过防火墙的过滤的。在新的规范中则允许同一 IIOP 连接上能够进行双向的调用请求，这样在某些严格的环境下不至于危及

*)"九五"国防科技预研项目《异种计算机系统互操作技术》资助，项目编号：15.1.2.1。曹晓阳　博士生，主要研究领域为开放系统技术，中间件及其应用。刘锦德　教授，博士导师。

维普资讯 http://www.cqvip.com

连接两端的安全。客户在建立连接时，指定其双向性；服务器对客户的回调不创建新的连接。

1.2 可互操作的名称服务[2]

将名称与对象引用联系起来称为名称绑定(Name Binding)；名称上下文(Naming Context)则是包含名称绑定集合的对象；名称解析(Name Resolved)是指在给定的上下文中确定名称所指的对象；绑定名称则是在指定的上下文中创建一个名称绑定。

在规范中 CosNaming 模块定义了名称服务所需的界面，它们是 NamingContext、BindingIterator 和 NamingContextExt 界面，NamingContext 界面定义了名称上下文的数据结构和其中的操作；BindingIterator 界面提供客户遍历名称绑定的操作；NamingContextExt 界面从 NamingContext 派生出来，提供使用 URL 格式和字符串格式名称所需的操作。

在规范中引入了新的 URL 格式名称，用来表示对象引用 iioploc 的 URL 格式类似于常用的 ftp 和 http 格式，用来指定远程的 CORBA 服务，包括名称服务。iiopname 的 URL 格式中包含表示名称绑定的字符串格式名称。字符串格式的名称使用"/"作为名称的分隔符，使用"."作为名称中 id 和 kind 域的分隔符。iiopname 调用远程的名称服务，解析字符串格式表示的名称，获得命名对象的引用。

例如，iiopname://1.1@myhost.xyz.com:9999/a/b/c 是符合该规范的 URL 格式，它表示在主机 myhost.xyz.com:9999 上的名称上下文，该上下文由主机上的名称服务代理通过 LocateReply 或 LOCATION_FORWARD 应答返回，该代理支持 IIOP1.1 版本规范，而 a/b/c 则用于在上下文中解析产生对应的对象引用。

2. 服务质量控制

在 CORBA 的原来版本中，对服务质量(QoS)的控制提及较少，而在网络环境中，系统的响应时间、操作优先权和系统容错等是应用中需要顾及的因素。在 CORBA 3 中，引入了几个与服务质量控制相关的新规范，它们是异步消息处理、最小化 CORBA、实时 CORBA 和 CORBA 容错等规范。

2.1 CORBA 异步消息处理[3]

原来的 CORBA 规范缺乏对异步消息处理的支持，在分布式应用中明显存在功能缺陷，因此在 CORBA 3 中定义了异步消息处理规范，其内容包括异步方法调用、与时间无关的调用和消息处理的服务质量等三个方面。

原来的 CORBA 中只提供了同步和延迟同步两种调用模式。在同步调用模式中，客户程序或线程在执行远程调用后将被阻塞，等待服务器返回结果；在延迟同步调用模式中，客户线程可以继续执行，但采用轮询方式查询是否有结果返回，而且该模式只用于动态调用界面。在 CORBA 3 中引入的异步消息处理提供了两种调用模式：回调和轮询。在回调模式中，客户将一个回调对象引用作为调用的一部分，服务器在返回结果时，激活客户端的回调对象。在轮询模式下，调用将返回一个对象，客户可以任意查询该对象获得所发出请求的状态。

与时间无关的调用提供了原 IIOP 所没有的存储转发功能，定义了扩展 GIOP 的路由协议。当被调用的对象未处于活动状态或无法激活时，调用请求可以暂时存储在代理中，在适当的时候转发给被调用的对象，请求的生命周期则可以由相应的服务质量控制来指定。同样，服务器对客户的应答也可以使用该存储转发机制。而扩展 GIOP 的路由协议定义了存储转发代理对象，它用于将消息发往下一代理或将消息同步分发给指定的目标对象，提供实现与时间无关调用所需的功能。

消息处理的服务质量在原 CORBA 规范中是没有提到的，它包括传输质量、队列管理和消息优先权等。客户和服务器可以通过设置消息优先权和生存时间，对时间敏感调用设置起止时间，或控制路由策略等方法，实现消息处理过程中的服务质量控制。提供对服务质量的控制，可以加强 CORBA 的可靠性和可扩展能力，使其能更适用于企业级的分布式计算环境。

2.2 最小化 CORBA[4]

最小化 CORBA 规范是主要针对嵌入系统而制定的，嵌入系统在完成后，通常是烧成芯片来进行批量生产，它要求在设计时就要确定可用的资源、要创建的对象和对象的位置。因此，用于嵌入系统的 CORBA 不需要动态机制，如动态调用界面和界面库等。最小化 CORBA 定义了 CORBA 的一个子集。

在最小化 CORBA 中舍弃了标准 CORBA 规范中的一些内容，这些内容在主流的 CORBA 应用中是有其存在价值的，但是它们也将耗费一定的资源和其它代价。在最小化 CORBA 中舍弃的这些内容，如果需要，可以在应用程序中来实现。

最小化 CORBA 规范中定义的内容保留了 CORBA 的主要优点，如应用程序的可移植性和 ORB 间的可互操作性。它适用于大多数的有限资源系统；同时与标准 CORBA 应用有完全的可互操作性；它支持全部的 IDL，只要有足够的资源，任何 CORBA 应用都可以在标准 CORBA 和最小化 CORBA 环境下运行。

最小化 CORBA 规范由于没有了标准 CORBA 所具有的动态特征，因此它可以在设计时进行界面类型

维普资讯 http://www.cqvip.com

检查，确定它的各项特征。

例如在符合最小化CORBA规范的ORB界面中，用于支持动态调用界面的creat_list和creat_operation_list操作被舍弃了，基本ORB操作所不需要的work_pending、perform_work和shutdown操作也被舍弃了；在Object界面中，用于界面库的get_interface操作和支持动态特征的is_a、non_existent和create_request操作都被舍弃。

虽然最小化CORBA舍弃了许多内容，但由于编译器和链接程序是按CORBA规范定义的，一些进一步的优化只能由应用程序来实现，这些内容包括类型安全、异常处理和继承等方面。应用程序如果不需要这些特征，可以不使用它们，以获得更精简的代码。

2.3 实时CORBA[5]

实时CORBA规范是基于CORBA2.2和CORBA消息处理规范对CORBA规范的扩展，它使用优先权模式将线程、协议和连接等资源进行标准化控制，以获得在实时环境中所需要的"可预见"特征。它将CORBA环境中的灵活性、可移植性和可互操作性引入了实时系统。

实时CORBA规范的核心是固定优先级调度。在实时系统中"及时性"与功能同等重要，控制实时系统中各组件的行为就可以提高整个系统的可预见性。实时系统通常不是纯粹的CORBA应用，它们不仅只有请求和应答信息的操作，还有处理I/O等其它与ORB无关的操作，规范称这些操作为"活动(activity)"。规范将活动抽象成传输中、静态和运行中三个状态，并提供界面使开发者可以定义和调整活动的状态。对于执行这些活动的线程，可以通过设定优先级来保证其运行的可预见性，优先级可以在客户端设定，也可以在服务器端申明。

实时CORBA包括两个新的模块，它们是RT_CORBA和RTPortableServer模块，所有的实时CORBA界面IDL定义都包含在这两个模块中。例如：RT_CORBA::RTORB定义ORB的扩展界面，它负责配置实时ORB的操作和管理其它实时ORB界面实例的创建和注销；RT_CORBA::Mutex界面提供对系统资源存取的并发控制；RT_CORBA::Priority界面定义了通用的与平台无关的优先级机制；RT_CORBA::Current界面提供存取CORBA线程优先级的方法；RT_CORBA::Threadpools界面提供对服务器端ORB上运行的线程进行管理。

2.4 CORBA的容错[6]

CORBA容错规范是对标准CORBA规范的扩展，它基于实体冗余、差错检测和故障恢复等方法来提高系统的可靠性。

实体冗余是指将对象复制，然后将复制的对象分布在不同的主机中，客户可以在它所要求的服务器出现故障时，将请求重新透明地发往复制的服务器对象。对象的复制可以由应用程序控制，也可以由CORBA的容错机制自动控制。差错检测和故障恢复则通过检查点(Checkpoint)设定和日志(Log)等方法来实现。

对象和它的复本被当作一个对象组(Object Group)来管理，对象的各个复本都有各自的对象引用，而整个对象组拥有一个可互操作的对象组引用(IOGR)。供客户使用的是该对象组引用，对象组象普通对象一样响应客户的请求，对象组内部的对象复制和故障恢复对客户是透明的。

在CORBA的容错规范中，还引入了容错域(Fault Tolerance Domain)的概念。一个容错域包括多个主机和多个对象组，一个主机可以支持多个容错域，容错域中的所有对象组都由同一个ReplicationManager创建和管理，但它们可以调用或被其它容错域中的对象调用。

在容错结构中，ReplicationManager，FaultNotifier和FaultDetector作为CORBA对象实现，ReplicationManager继承了PropertyManager，ObjectGroupManager和GenericFactory界面。虽然在一个容错域中只能有一个ReplicationManager和一个FaultNotifier对象，但是它们可以象应用程序对象一样被复制来提高可靠性。Factory和FaultDetector对象在各主机中都存在，它们不需要被复制。此外，消息处理、日志和故障恢复机制在各主机中都应存在，它们不是CORBA对象，而是ORB或ORB与操作系统之间的一部分。

用户通过PropetyManager定义对象组的容错属性，如对象组和对象组成员的一致性状态是由容错结构控制还是由应用程序控制。通过GenericFactory的create_object()方法创建对象的复本。通过ObjectGroupManager的create_member()、add_member()和remove_member()方法控制对象组成员的添加、删除和定位。在各主机上运行的FaultDetector对象调用PullMonitorable中的is_alive()方法来监视本机应用对象，全局的FaultDetector对象负责监控各主机本地的FaultDetector对象，在由支持复制的各应用对象所实现的Checkpointable界面中，get_state()方法提供容错所需的日志机制，set_state()方法则提供故障恢复机制。

3. CORBA组件[7]

在CORBA 3中最引人注目的就是新引入了CORBA组件的技术规范。组件技术的引入对程序设

维普资讯 http://www.cqvip.com

计者和用户都会带来好处，如使用组件技术可以简化软件的开发过程，降低软件的运行维护费用，提高软件的可重用性。

CORBA 组件技术是服务器端的技术，它包括以下三个方面：

· 组件容器，它是 CORBA 组件的运行环境，提供事务处理、安全性、持久性和事件等基本机制；

· 与 Enterprise Java Bean 集成；

· 软件分发格式，使 CORBA 组件可以市场化。

3.1 组件模型

组件对于 CORBA 是新的类型，扩展了对象的定义，它可以在 IDL 中指定，也可以存在于界面库(Interface Repository)中。指示组件的组件引用，是特殊的对象引用，组件的定义则是界面定义的特殊化和扩展。

组件可以提供多个对象的引用，可以支持不同的 CORBA 界面，这些引用称为面(facets)。而每个组件有一个与它定义相对应的界面，称为组件的等价界面，客户可以使用该等价界面遍历组件的各个面。

组件的实例标识主要使用它的组件引用，其次是它的面引用集合，此外组件还有与它相关的主键值，主键值对客户是可见的数据，客户可以使用它来获得组件实例的引用。组主键值不是组件的特征，它由管理组件的 home 来维护。

在规范中定义的组件 home 是用来管理某指定类型组件实例的界面，它可以管理组件的生命周期，以及组件实例和主键值之间的对应关系，home 的定义中必须指定唯一的可管理组件类型，相同类型的组件可以由多个不同的 home 管理，但一个组件实例只能由一个 home 管理。

组件技术的引入，需要扩展 IDL，组件在 IDL 中使用新的关键字 component 来定义，使用 provides 声明界面，emits 或 publishes 声明事件的源，consumes 声明事件的处理过程，使用 attribute 声明属性。组件的面使用关键字 object 和 provide_(name)定义。

在规范中还引入了组件实现定义语言(CIDL)，是用来描述组件实现结构和状态的描述性语言。支持组件的 ORB 产品能根据 CIDL 生成组件实现的框架，组件的编写者扩展这些框架完成组件的实现。在 CIDL 中包括用来声明组件状态集合的结构(称为存储类型 storage type)，和管理这些结构的界面(称为 storage homes 和 persistent stores)。这些结构和界面可以用来实现组件的持久性。

3.2 组件容器

组件容器提供 CORBA 组件运行时的执行环境，是基于 ORB、POA 和一些 CORBA 服务的服务器端框架。

组件容器对象的各界面在 Container 模块中定义。组件容器包含在组件服务器进程中，一个组件服务器进程中可以包含多个组件容器。组件容器的创建和撤消由容器管理器(Container Manager)来完成，创建容器时通常同时创建所需的 POA。POA 用于创建可供客户使用的组件引用，以及在收到客户请求后激活组件实例。

通过容器的 Container Type 可以指定要使用的 CORBA 服务，通过 CORBA::ORB 界面中的 resolve_initial_reference 操作获得所需服务的初始引用，并完成服务所需的配置操作，如：建立事件源和处理程序所需的通道，创建和初始化持久操作所需的数据库连接，确定用于解析组件名称的上下文。

在 CORBA 组件规范中，组件容器分为五类，它们是：Empty 容器(管理的组件其实现可以使用 CORBA 中的所有界面)、Service 容器(管理无状态的 CORBA 服务组件)、Session 容器(管理有状态的 CORBA 会话组件)、Process 容器(管理使用持久化机制封装了服务器上所有数据的进程组件)和 Entity 容器(管理使用持久化机制封装了服务器和客户之间共享数据的实体组件)。

3.3 与 EJB 集成

基于 Enterprise Java Bean 规范的 EJB 是一种 Java 组件，根据 CORBA 规范的定义，EJB 与 CORBA 组件是兼容的。分布式环境中的应用程序可以混合使用在 CORBA 组件服务器中运行的 CORBA 组件和在 EJB 环境下运行的 EJB，CORBA 组件服务器既能支持 CORBA 组件也能支持 EJB，按 EJB 模式使用 Java 编写的 CORBA 组件可以部署到 EJB 服务器中。

部署在 CORBA 组件服务器中 EJB 对于其它的 EJB(无论是否部署在 CORBA 组件服务器中)都可以被看作是 EJB 远程界面，通过 RMI/IIOP 可以访问 CORBA 组件服务器中的 EJB。

基于 Java 到 IDL 的映射和 IDL 对组件的扩展，通过 RMI 编译器可以产生组件形式的 EJB 封装或代理，很容易地实现 EJB 在 CORBA 组件环境中的应用。

EJB 可以有两种方法部署到 CORBA 组件服务器中，一种是将 CORBA 组件容器设计成 EJB 容器的超集，包含 CORBA 组件容器和 EJB 容器的内容及所需的协议，称为 Direct Hosting；另一种方法是创建一组适配器对象，使 EJB 看起来象 CORBA 组件，称为 EJB Adaptation。在 Direct Hosting 方法中，如果 ORB 与容器都使用 Java 编写，则部署既简单又直接；如果 ORB 与容器不是使用 Java 编写，则容器的 EJB 界面需要有 Java 代理，EJB 也必须有编译的语言代理。在 EJB

维普资讯 http://www.cqvip.com

Adaptation 方法中，需要用 Java 实现 EJBObject 和 EJBHome 界面。

3.4 组件的打包和部署

组件的实现可以被打包和部署。CORBA 组件包由某个抽象组件的一个或多个实现组成，它包括一个描述符和一组文件。组件包可以安装到计算机上，或与其它组件组成集合(assembly)，包作为集合的一部分来部署。

组件包通常作成文档文件，以".car"为文件扩展名。组件包中的描述符用来描述包的特征和指向包中其它的文件，通常使用".csd"作为扩展名，XML 被作为描述包中元素的语言。

在组合成集合或部署时，需要使用组件描述符文件中描述的组件信息。组件描述符文件通常使用".ccd"作为扩展名。组件描述符分两个主要部分，第一部分是描述组件的一般信息和部署时所需的组件特征；第二部分是描述组件所支持的界面、继承的组件、使用和提供的端口等。组件描述符由 CIDL 编译器产生，用户可以手工修改和完善。与包描述符一样，组件描述符也使用 XML 作为描述语言。

一组相关的组件包组成组件集合，可以用来同时分发多个组件。组件集合也是被作成文档文件，以".aar"为扩展名。组件集合使用一个集合描述符文件来描述集合，包括集合中包含的组件、连接信息和分区信息等。集合描述符文件使用".cad"为扩展名，使用 XML 作为描述语言。

部署工具用来在安装站点上部署单独的组件和组件的集合。部署的过程可以分为四个步骤：

- 用户使用部署工具确定各组件安装到哪台计算机，同一集合中的组件可以安装到一台计算机上，也可以分布到网络中；
- 在各平台上安装组件实现，部署组件实例。如果在主机上已安装有相同唯一标示符(UUID)的组件，则不再安装；
- 在指定的主机上演示组件；
- 根据集合描述符中的描述，连接各组件。

基于组件的应用程序，还可以使用脚本语言控制组件完成应用程序的功能。这样不需要熟练的程序员就可以修改和调整应用程序的运行逻辑[8]。因此，在 CORBA 3 中还引入了针对组件的 CORBA 组件脚本规范，使得 CORBA 组件的应用更加方便。

结束语 自 1991 年 OMG 推出 CORBA 的第一个版本以来，CORBA 技术在面向对象的分布式系统领域中不断完善和发展。最新的 CORBA 3 规范引入了几种新的技术，虽然其中也存在着不足和需要完善的地方，但我们仍然可以看到 CORBA 旺盛的生命力，及其在分布式领域强大的竞争力。

参考文献

1 OMG CORBA/Firewall Security. http://cgi.omg.org/cgi-bin/doc?orbos/98-05-04
2 OMG Interoperable Name Service. http://cgi.omg.org/cgi-bin/doc?orbos/98-10-11
3 OMG CORBA Messaging. http://cgi omg org/cgi-bin/doc?orbos/98-05-05
4 OMG minumum CORBA. http://cgi.omg.org/cgi-bin/doc?orbos/98-08-04
5 OMG Real-time CORBA. http://cgi.omg.org/cgi-bin/doc?orbos/99-02-12
6 OMG Fault tolerance CORBA. http.//cgi.omg org/cgi-bin/doc?orbos/99-10-05
7 OMG CORBA Components. http://cgi.omg.org/cgi-bin/doc?orbos/99-02-05
8 OMG CORBA Component Scripting http://cgi.omg.org/cgi-bin/doc?orbos/99-08-01

(上接第 15 页)

4 Vckovski A. Special Issue·Interoperability in GIS. International Journal of Geographical Information Science, 1998,12(4)
5 Kottman C. Geodata Interoperability: What does It Mean for Business Geography? Available at: http://www.OpenGIS.org/
6 Chen Shupeng. Geo-Information and Regional Sustainable Development. Surveying and Mapping Publication, Beijing, 1995
7 Open GIS Consortium. The OpenGIS Abstract Specification. OpenGIS Project Documents, 1999. 99～116
8 Open GIS Consortium. OpenGIS Simple Features Specification For OLE/COM. Revision 1.0.18 May, 1999
9 Open GIS Consortium. OpenGIS Simple Features Specification For CORBA. Revision 1.0.18 May, 1998
10 Open GIS Consortium, OpenGIS Simple Features Specification For SQL. Revision 1 1.5 May, 1999
11 Open GIS Covsortium. OpenGIS Web Map Server Interface Implementation Specification. Revision 1.0.0.19 April, 2000
12 Open GIS Consortium. Coordinate Transformation Services. Revision 1.00.12 January, 2001
13 Open GIS Consortium. Grid Coverage. Revision 1.00.12 January, 2001
14 Open GIS Consortium. OpenGIS Web Map Server Interface Implementation Specification, 19 April, 2000
15 Open GIS Consortium. Geography Markup Language (GML). 20 February, 2001
16 Open GIS Consortium. The OGC Technical Committee Technology Development Process. Wayland, Massachusetts, 1997
17 ESRI. ESRI Open Strategy—White Papers on SDE/CAD Client & Spatial Data Warehousing. 1998. Available at: http://www.esri.com/base/company/opengis/
18 Goodchild M. Future Direction for Geographic Information Science. Geographic Information Science, 1995(1)