计算机科学1999Vol. 26№. 10

实时 CORBA

Real-time CORBA

骆志刚 刘锦德

（电子科技大学计算机学院 成都 610054）

Abstract Distributed real-time system may be across several heterogeneous platforms. Application development is simplified and cooperation between heterogeneous platforms is easily implemented by using CORBA. There is increasing demand to extend CORBA to support real-time application. Distributed real-time system architecture based on CORBA is presented in the paper. Real-time CORBA architecture is discussed in detail and the key components that needed to be optimized are given.

Keywords Distributed real-time system, CORBA, Cooperation, Architecture, Optimization

1 引言

随着计算机硬件、软件的迅速发展，尤其是网络的广泛使用，大量的应用需要跨网段、跨平台地进行协同工作，分布式实时系统也面临同样的问题。事实上，典型的分布式实时系统必须经过精心定制以便能在专用系统上运行。虽然这些定制的系统为实时应用的操作提供了所需条件，但是它创建了一个不灵活的结构，存在不容易修改、升级和与第三方产品集成的缺点。

另外，支持具有时限操作的需求出现在不同于传统实时系统的系统中。具有家用娱乐系统的家用计算机需要能够保证网上数字视频和音频的质量。实时需求已经开始进入传统的办公计算机系统，通过网络方式，能进行视频会议、网上教学等多媒体应用。这些家用系统和办公系统更多的是传统的 CORBA 系统的应用域。

当前的 CORBA 结构并不适用于具有严格时限的应用，缺乏对实时的支持，但是 CORBA 为分布式实时系统提供了许多好处：开发简单性、可互操作性、灵活性、可维护性、可重用性等，这些优点都是当前其它分布式实时系统无法提供的。CORBA 提供了真正的通用软件总线结构，它可以克服当前分布式实时系统的不灵活性，因此 CORBA 与实时系统的结合自然成了今后分布式实时系统的一个重要发展方向。

本文给出了基于 CORBA 的分布式实时系统的结构，指出当前 CORBA 在支持实时方面的不足，着重讨论如何扩展 CORBA 结构及如何优化以便使之支持实时应用。

2 基于 CORBA 的分布式实时系统结构

分布式实时系统在逻辑上可抽象为：由域和互联两者组成，如图1所示。互联在物理上是以数据通信设备和局域网，甚至广域网实现；而域则由端系统组成。

为了确保整个系统的全局实时性，图1中的各个组成部分都应当是实时的，即它们分别采用实时域和实时互联，因此，应关注两个问题：端的实时性和网络实时性。

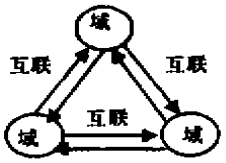

图1 分布式实时系统的抽象

基于 CORBA 的分布式实时系统的结构如图2所示，它由四个部分组成：实时应用、实时 CORBA、操作系统、网络通信，其中前三者构成了端系统。

对于非专用的实时系统（如家用系统和办公系统），操作系统可以选择具有一部分实时特征的操作系统，如 Solaris 和 Windows NT。若是专用系统，对时限要求极高（如航空控制和武器控制）的系统，则应当选择专用的实时操作系统。

网络的实时性在当前的网络状况下，特别在广域网中，要取得满意的效果还很困难。各种网络设备的性

骆志刚 博士生；刘锦德 教授，博士生导师。

能参差不齐以及传统网络协议本身的局限性(如不支持服务质量),要使整个网络都支持实时性还有很大的困难。但在局域网中,负载不重的情况下,网络的实时性是可以满足的。

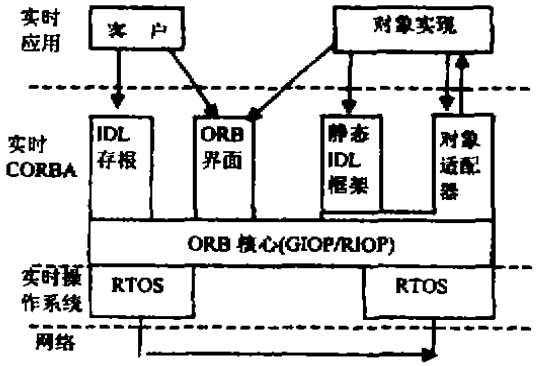

图2 基于CORBA的分布式实时系统结构

用CORBA构造分布式实时系统时,必须解决一个重要的问题:如何扩展CORBA结构才能支持实时应用,本文将主要从实时应用的要求出发,进行讨论。

3 CORBA在实时应用中的局限性

在实时系统中,时间是一种重要的资源,对外部事件的响应和任务执行都必须在限定的时间内完成。在分布式系统中,还必须在限制的时间内完成消息的发送和接收。实时系统中,输出结果的正确性不仅取决于计算所形成的逻辑结果,还要取决于结果产生的时间。OMG在制订CORBA规范时,首先考虑的是互操作性,用以解决异种平台的不同对象的协作问题,而未对时限作过多的考虑。所以,当前CORBA特别适合传统的请求/应答模式,而要将其应用于实时领域中还存在以下的局限性:

1)CORBA结构上的局限性:对于实时系统,操作按优先级驱动或由事件驱动,执行有其时限,应该尽量避免优先级反转和非确定性的发生。当前CORBA没有定义优先级,事件服务不适用于实时应用(无事件优先级,无事件过滤,无法处理事件相关性),也没有定义适用于实时系统的调度,这就从结构上无法避免优先级反转和非确定性的发生。

2)缺乏服务质量(QoS)的支持:当前的CORBA,不提供端到端的QoS支持。对客户来说,无法向ORB表达请求的优先级,也无法通知ORB操作的执行时限。对服务方来说,都统一按先进先出调度,无法根据QoS分配资源,也无法根据资源的消耗情况作出操作调整,甚至拒绝操作,例如一视频服务器访问的客户过多时,应当能够拒绝更多的客户请求;带宽紧张时,可以损失图象的质量,尽量保证声音的质量。

3)缺乏性能优化:当前的CORBA没有过多考虑性能的问题。在整个实现中,存在大量的数据转换(masrshal/unmarshal)和数据拷贝,这些操作极为消耗资源,时间代价很大,优化的意义就显得更大。消息在客户和服务器之间传递,要经过多层,每一次的分解(demultiplexing)和分派(dispatching)也很耗时,也需要优化。

4)缺乏实时应用编程支持:实时应用,需要更多的应用编程支持,进行实时控制,来完成具有时限的操作。

由于CORBA在实时方面的支持不够,这就需要从结构上加以扩展,从实现上加以优化,使之能用于构筑分布式实时系统。

4 实时CORBA的结构

实时CORBA主要从结构上对CORBA进行扩展:(1)避免优先级反转和非确定性的发生;(2)解决可调度实体的资源竞争问题;(3)限制操作延迟。

实时CORBA中,将一个线程作为一个可调度实体,它代表一个结点内的控制流的序列。在实现上它直接调用系统提供的线程,用户可以通过编程界面直接对可调度实体进行操纵。与可调度实体相对应的是"活动",它被定义为可以跨越系统边界的控制流的序列。活动的使用主要是用于抽象地描述分布实时系统中优先级的传递。活动的生命周期可以按用户的要求改变。通常,一个活动起始于客户发动调用的线程的开始执行时,终止于该线程的结束时。

4.1 优先级定义及其使用策略

实时CORBA中定义了一个通用的与平台无关的优先级格(scheme),称为CORBA Priority,主要是解决异质环境中优先级表示的差异,使实时CORBA应用能够用一致的策略去处理不同平台的不同优先级格,即实时CORBA应用在不同系统中使用CORBA Priority表示优先级。

实时CORBA的可调度实体是线程,每个线程要与某优先级绑定。对于可调度实体,通过ServerPriorityModePolicy确定其优先级。优先级的确定策略有两种:CLIENT_PRIORITY_PROPAGATION和SERVER_SET_PRIORITY。前者表示优先级来自CORBA调用,即随客户请求传递而来,以使后来完成调用的线程在这一优先级上运行;后者表示执行客户调用的线程的优先级由服务器确定。

线程执行过程中的优先级必然是本地优先级(native priority),这就存在CORBA优先级和本地优先级之间的映射。实时CORBA定义了两者的映射:to_native()和to_CORBA,前者表示CORBA优先级到本地

优先级的映射，后者表示本地优先级到 CORBA 优先级的映射。

4.2 互斥

互斥是控制可调度实体对共享资源进行串行访问的同步机制。实时 CORBA 中定义了互斥界面，解决资源的竞争问题。互斥提供了加锁和解锁操作。

实现互斥对象时，需要提供优先级继承协议，以解决资源竞争中的优先级反转问题。当一个线程在一个受互斥对象保护的一个区域内运行，若有一个线程的优先级更高，则资源被强占。

4.3 服务方的线程策略

在服务方，通过多线程机制并发完成客户的请求，它通过线程池(thread pool)管理服务方执行的线程。线程池提供了下列特征：

线程的预分配：通过事先分配足够多的线程，用以满足一定数量的并发调用，这有助于减少优先级反转，减少延迟和增加可预测性，避免了调用时的线程的析构和重建。

线程池分区：将一个线程池按不同的优先级范围分区，将不同的分区与不同的 POA[1]相关，不同优先级的调用使用不同优先级的分区，也有利于减少优先级反转。

线程使用的限制：通过线程池，能限制一个 POA 可以使用的线程的数量，也就可以在一个系统中限制线程的使用，它可以与线程池分区一起避免优先级反转。

线程池的创建有两种类型，不分区和分区：Create_ThreadPool 和 Create_ThreadPool_With_Lane。对于分区的线程池，当优先级高的线程池分区用尽时，若参数 borrow 设置为真时，可以借用优先级较低的线程池分区。

线程池策略可以用于 POA 级和 ORB 级，一个 POA 只能使用一个线程池。线程池策略用于 ORB 时，ORB 创建 POA 时，指定一个缺省线程池给 POA，POA 可以改用其它线程池。

4.4 客户显式连接策略

在通常的 CORBA 应用中，一般使用隐式连接策略，即在调用操作过程中，客户才与服务方建立连接，这不适用于实时系统。实时 CORBA 中，客户需要使用显式连接策略，即在操作调用前，首先与服务方建立连接，这样减少了操作调用过程中的连接延迟。

为了减少因为使用非优先级的传输协议而造成的优先级反转，实时 CORBA 为客户提供了通过多连接的通信设施，每一个连接处理一定范围内的优先级调用。

4.5 实时 Inter_ORB 协议(RIOP)

无论是 GIOP 还是 IIOP 都没用详细定义端到端的 QoS 的需求，这使得 GIOP/IIOP 不适用于分布式实时应用，因为它们不能处理延迟和抖动的问题。对于实时系统，为了保证端到端的可预测性，需要支持 QoS 的协议 RIOP。RIOP 是 GIOP 的映射，它允许端到端通信过程中传递 QoS。RIOP 可以是 GIOP 到不同网络的映射。

尽管 RIOP 支持 QoS，由于网络的复杂性、传统网络协议的局限性，很难保证网络的实时性，尤其是广域网的实时性。但在负载不太重的局域网中，可以保证。RIOP 运用在 ATM 网中，它将 RIOP 的 QoS 映射为 ATM 的 QoS，实时性是可以保证的。

4.6 调度服务

在实时 CORBA 中，调度服务使用实时 ORB 的原语跨越实时 CORBA 系统来实现各种固定优先级的调度策略。通过这一方法，应用中的低层实时构造的复杂性得以屏蔽。调度服务可以在整个基于 CORBA 的分布式实时系统中实现统一的调度策略，如全局比率单调调度策略(Rate Monotonic Scheduling)。调度服务的实现选择 CORBA 优先级，POA 策略，优先级映射来实现统一的调度策略。不同的实现可以提供不同的调度策略。

调度服务使用 CORBA 活动和 CORBA 对象的名字作为参数，在其内部将名字映射为具体调度策略，这一抽象有助于提高可移植性。

4.7 实时事件服务

CORBA 的事件服务通过事件通道将事件提供者和事件消费者分离(如图3a 所示)，为分布式系统提供了灵活的异步通信和组通信模式。但是要将其用于实时系统中，还必须增加实时事件的调度和分派，周期任务处理，事件过滤和事件相关性的处理。

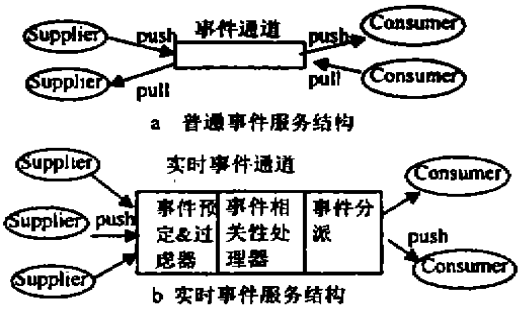

图3 事件服务结构

1) 可移植对象适配器、有关 POA 和后面提到的 Servant、请参见 CORBA 规范[2]中的 POA 部分

实时事件服务中的事件通道如图3b所示，其中，消费者通过事件预定和过滤器可以预定消费者需要的事件，事件还可以过滤，去掉消费者不关心的事件；通过事件相关处理器，可以处理事件的相关性；最后通过分派模块（与调度服务相合作）、向消费者传递事件。

在普通事件服务中，可以使用推（push）和拉（pull）操作，针对实时系统的事件驱动方式，只需要推操作。

5 性能优化

上一节主要从结构上讨论了CORBA在结构上的扩展，以便适用于实时应用。基于CORBA的分布式实时系统利用了CORBA解决异种平台的成熟技术，但由于中间增加了一层，必然会造成性能下降的问题，而在实时系统中，时间是一种重要的资源，因此应用于实时系统的CORBA的性能的优化尤为重要，应尽量减少延迟。在实时CORBA实现过程中，应当着重注意以下几部分的优化：

1）数据转换的优化：在实时系统中，一般使用静态调用，因为动态调用太耗时。对于客户方的存根（Stub）和服务方的框架（Skeleton），要进行大量数据的封装和解封装。通常可使用编译型和解释型两种方案，对于一般的实时系统，使用编译型方式，尽管内存消耗大，但其效率高。而对于嵌入式实时系统，内存有限，只能采用解释型方式，其优化就更为重要。

2）内存使用优化：调用从客户到服务对象，要经过多层（Stub，I/O系统，网络适配器，对象适配器，Skeleton），数据需要多次的拷贝。每一次都会消耗大量的资源，如：CPU，内存等。内存管理的优化就显得十分必要。在实时系统中，使用零拷贝机制，使动态内存分配和数据拷贝最小化。

3）请求的分解和分派优化：客户的请求到达服务方所在的主机后，还要经过多层次的分解和分派，才能找到合适的操作（如图4所示），其过程如下：

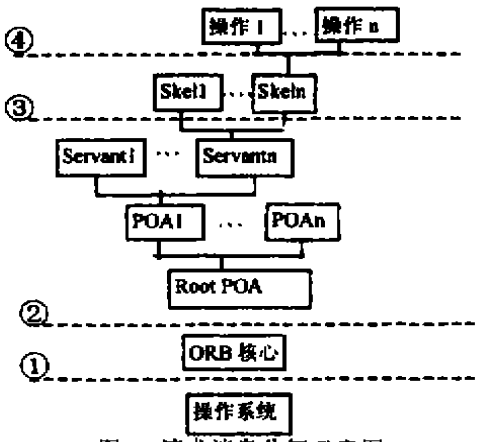

图4 请求消息分解示意图

①操作系统收到请求消息，传递给ORB核心。

②ORB核心利用请求消息中的对象键（Object Key）找到合适的POA和Servant，这一过程可能包含多次重复，因为POA可以是多层次结构。

③Servant利用操作名找到对应的框架，框架将请求消息解封装为操作调用参数。

④分派操作，将参数上调给操作，操作执行。

分层的优点在于设计和实现容易，每一层的改变不会影响其它层，但同时也有其缺点，那就是增加开销，效率降低，增加了优先级反转和非确定性的可能性。

请求的分解和分派优化中，主要是改进搜索算法，根据不同的特点使用不同的算法，例如：②可以使用活动（active）分解法；③可以使用哈希表法。

结束语 从传统的各种控制系统，到新兴的Internet和Intranet多媒体服务，都需要实时的支持，大量信息的处理，需要多机合作，这使得分布式实时系统成为必要。平台的多样性和差异性，必然需要分布式实时系统克服异种平台的差异，进而进行协同工作，而这一方面正是CORBA的特长。利用CORBA构筑分布式实时系统，降低了软件开发的费用，增加了应用的可互操作性、可重用性、可扩展性和可维护性。针对CORBA在实时处理方面的不足和实时系统的要求，本文讨论了CORBA结构的扩展，使之适用于实时系统。同时，本文还给出了实时CORBA在实现过程中，应当重点优化的部分。基于CORBA的分布式实时系统是分布式实时系统的一个新兴分支，它将是下一代CORBA需要解决的问题之一。

参考文献

1 Object Management Group. Resl-time CORBA1.0 Joint Submission. OMG Document orbos/98-12-05 ed., December 1998

2 Object Msnagement Group. Common Object Request Broker: Architecture and Specification, 2.2 ed., February 1998

3 Stankoric, Misconceptions about Real-time Computing. IEEE Computer, 1988, 21

4 Shin K. HARTS: A Distributed Real-time Architecture. IEEE Computer, 1991, 24(May)

5 Schmidt D C, et al. The Design and Performance of Real-time Object Request Brokers. Computer Communication, 1998, 21(April)

6 Ramamritham K, et al. Distributed Scheduling of Tasks with Desdlines and Resource Requirements. IEEE Trsnsactions on Computers, 1989, 38(August)

7 Schulzrinne H, et al. RTP: A Transport Protocol for Real-Time Application. RFC1889. Internet Engineering Task Force(IETF), 1996(Jan.)

8 Zhang L, et al. RSVP: A New ReSerVation Protocol. IEEE Network, 1993(Sept.)