CORBA 嵌入式计算机系统 (25)

计算机科学 2000Vol. 27№. 12

101-104 互操作

嵌入式 CORBA 模型的研究

The Research of Embedded CORBA Model

TP338

郭 兵 熊光泽 刘锦德 黎忠文

(电子科技大学计算机科学与工程学院 成都 610054)

Abstract Embedded system increasingly demands interoperability across heterogeneous platforms, it becomes necessary to introduce CORBA model into embedded system. This paper first presents the Embedded CORBA model, then analyzes the limitation of Real-time CORBA specifications, last emphasizes the structure of Embedded CORBA model and its three main implementation manners.

Keywords Embedded system, Distributed embedded system, Distributed computing, CORBA, Embedded CORBA, Interoperability

1 引 言

嵌入式计算机系统(下称嵌入式系统)是实时系统的重要组成部分,其结构紧凑、资源有限,一般由嵌入式微处理器等硬件及其软件(包括实时操作系统RTOS和实时应用程序)组成,具有嵌入性和实时性等特点。它通常以SOC(system on chip)、单片机、单板机、多板式箱体结构、嵌入式PC等形式嵌入到各式各样的设备或大系统(如数字移动电话、路由器、导弹)中,作为设备或大系统的处理和控制核心。嵌入式系统的运行需要一个RTOS的支持,这是它不同于一般单片机或单板机应用的关键之处。随着Internet和后PC时代的到来,嵌入式系统的应用范围日益广泛,从工业控制系统到家用消费电子、从Internet装置到手持移动计算装置[4],大量信息的处理,需要嵌入式系统跨平台地进行协同工作,即成为分布嵌入式系统,因而对互操作分布式处理的需求变得日益迫切。

CORBA模型OMG(对象管理集团)推出的分布式对象计算模型,目前在由台式机和服务器平台构成的大范围异构环境下(如电信和金融领域)实现互操作已取得成功。因此,将CORBA模型引入嵌入式系统中,扩展为嵌入式CORBA模型,实现嵌入式系统与台式机和服务器之间更大范围异构环境下的互操作成为必要[3]。

2 实时CORBA规范的局限性

目前OMG推出的实时CORBA1.0规范[6]是CORBA2.2规范和消息规范[5]的扩展,对于固定优先级的CORBA应用程序,支持端到端的可预测性。随后推出的动态调度规范[8]是实时CORBA固定优先级调度的扩展,对应用程序调度服务增加动态调度特性,实时CORBA通过定义标准界面和QoS(服务质量)策略,允许应用程序指定端到端的QoS,它主要在以下两个方面扩展CORBA规范以支持端系统的实时性:

1)从结构上扩展CORBA规范。实时CORBA的扩展结构包括RTCORBA::PriorityMapping、RTORB、RTCORBA::Threadpool、RTPOA、RTCORBA::Current等五个部分,实现优先级定义及映射策略、RTCORBA::Mutex互斥界面、客户方显示连接策略、服务方线程策略、RIOP协议、实时调度服务、实时事件服务等功能[2],避免优先级反转和非确定性的发生。

2)对CORBA的性能进行优化,包括数据转换优化、内存使用优化、请求的分解和分派优化等,尽量减少操作延迟。

实时CORBA 1.0规范面向典型实时系统,没有针对嵌入式系统的特点,因而应用在嵌入式系统中,存在一定的局限性,其局限性主要表现在:

1)缺乏硬实时支持。对于嵌入式系统而言,实时性是考虑的首要因素。实时性指标包括响应时间和确定性。实时性按响应时间分为硬实时和软实时两类。在硬实时系统中,要求的响应时间和处理时间既短而又确定,如果超出规格所要求的时间,所产生的结果就不可用,软实时系统对于响应时间和处理时间的要求较低,

郭 兵 博士研究生,主要研究方向为嵌入式系统软件开发环境和中间件技术。熊光泽 博士生导师,主要研究方向为实时计算机系统及其应用。刘锦德 博士生导师,主要研究方向为开放系统、分布式实时系统。黎忠文 博士研究生。

如果时间超出，仅仅是不符合规格，所产生的结果仍然是可用的。当前实时CORBA模型重点解决端到端的可预测性(即确定性)，而对ORB端系统需要的硬实时特性，实时CORBA模型尚未提供相应的机制和策略。

2)缺乏可嵌入性。嵌入式系统为嵌入式应用的需要，其结构紧凑、资源(主要为内存)有限。实时CORBA规范作为CORBA2.2的扩展，其组成结构仍是面向通用系统，大而全，对于资源限制问题未作考虑，虽然能够选择传送层协议以及设置协议属性，但对实时CORBA的功能部件却不能进行灵活的裁剪和配置。

3)高互操作性和实时性可能产生冲突。对实时CORBA的某些性能优化，可能产生新的非确定性和抖动，如用自组织的搜索结构来分解客户端请求，提高互操作性能，这会降低给定操作的可预测性。因此，进行性能优化时，应采用合适的算法和数据结构，兼顾高互操作性和实时性两个指标，如可通过减少过度搜索和避免优先级反转提高客户端请求分解的性能和可预测性。

4)没有考虑嵌入式CORBA应用程序的交叉开发结构。嵌入式软件的开发方法是，在宿主机上建立开发环境，进行应用程序编码和交叉编译，然后宿主机同目标机连接，将应用程序下载到目标机上进行交叉调试，应用程序经过调试和优化，证明不存在逻辑错误，同时确定性满足需求后，最终将应用程序固化到目标机中运行。嵌入式CORBA应通过提供目标机ORB运行函数库和基于宿主机的开发工具(包括实时IDL编译器、实时命名服务器等)，以适应嵌入式软件的交叉开发方式。

3 嵌入式CORBA模型的结构

嵌入式CORBA模型作为实时CORBA规范的进一步扩展，在保证CORBA2.0规范兼容性和保持实时CORBA优点的基础上，为实现嵌入式CORBA的强实时性和可嵌入性，应重点改善以下几个方面：

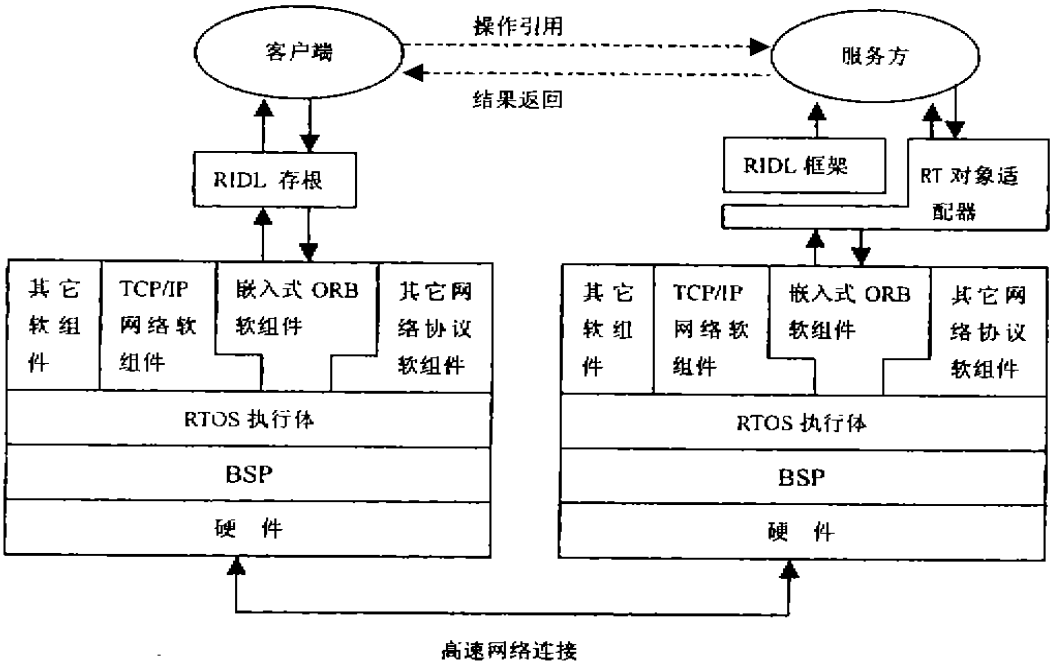

图1 嵌入式ORB的体系结构

1)实时CORBA规范的裁剪。依照Minimum CORBA规范[5]，对实时CORBA规范进行裁剪，在保证实时性的前提下，使之仅包含一个精减的ORB核规范，满足嵌入性的要求。

2)嵌入式ORB端系统的一体化设计方法。嵌入式ORB端系统的性能很大程度是一个实现问题，因此，它的设计方法至关重要。嵌入式ORB端系统，通常包括静态和动态的实时调度、实时请求的分解和分派、实时事件处理、实时I/O子系统、RTOS、实时ORB核连接和并发控制结构等部件，相互间紧密配合。首先，能够对部件进行裁剪，如选用静态实时调度还是动态实时调度，其次，能够对每个部件的属性等诸多参数进行配置，如线程池预先创建的线程数及其使用策略，以更好满足嵌入性的要求。

3)嵌入式ORB端系统的组件化体系结构。相对于台式机和服务器而言，嵌入式系统是比较专用和特殊

的系统，有其特定的工作环境。其应用程序是RTOS执行体和应用程序一体化的程序，二者结合紧密，RTOS通常采用基于优先级、可完全抢占的调度策略、微内核设计，操作系统的其它功能（网络、GUI、JVM、文件系统等）以软组件形式提供，易于裁剪和配置[1]。因此，嵌入式ORB应作为RTOS的一部分，以软组件形式提供（如图1所示），同底层RTOS执行体和网络QoS集成为一体[2]，减少ORB实现中调用的层次，充分利用RTOS的硬实时机制，实现嵌入式ORB的硬实时特性。

4）表示层的优化。对于客户端的存根和服务方的框架，IDL编译器通常可使用编译型和解释型两种方案。对于嵌入式系统，解释型同编译型比较结果显示[3]，解释型的性能通常是编译型方式的75-100%，而解释型存根和框架的代码尺寸分别是编译型的26-45%和50-80%，可见，解释型存根和框架对于资源紧张的系统是首选，但对响应时间无疑是有影响的。

4 嵌入式CORBA模型的实现方式

嵌入式CORBA的实现方式主要有以下三种：

4.1 Minimum CORBA

OMG已经采纳了一项最小化CORBA的标准，即Minimum CORBA规范[5]，是CORBA规范的子集。它是从CORBA2.2规范裁剪而来，包含一个精减的ORB核规范，面向嵌入式系统和其它资源（如内存）有限的系统。它有以下几个主要特点：

1）移去动态设施，包括界面仓库（Repository）、实现仓库、动态调用界面、动态框架界面，同时与这些动态设施相关的操作（如create_list）都要从ORB和IDL中移去，

2）完全的IDL兼容性，但由于移去动态设施，有些数据类型可能没有意义（如Context）。

3）不提供运行时安全服务，需要用户设计应用时考虑安全问题。

4）其它一些特性，如不支持"any"类型码；用户异常和系统异常可以删减，由应用程序处理异常；当应用程序没有使用IDL多继承时，可以删减多继承机制等。

可见，Minimun CORBA仅是现行CORBA 2.0规范的裁剪，并非实时CORBA规范的扩展，满足了嵌入式系统资源限制的要求，但未满足嵌入式系统的硬实时性和可配置性，目前还没有符合Minimum CORBA规范的产品推出。因此，该规范仍需继续发展和完善，才能更好满足嵌入式系统的需求。

4.2 IIOP引擎

IIOP引擎也是CORBA2.0规范的子集，是一些ORB厂商（包括Sunsoft、Orbix、Highcomm等）提出的嵌入式CORBA解决方案，目前已经推出了相关产品，如Orbix公司推出的IIOP Engine for Vxworks和I-IOP Engine for QNX，这些产品仅部分遵照Minimum CORBA规范实现，但保证了最大程度的CORBA2.0规范兼容性和对象间完全的互操作性。二者主要区别是：

1）Minimum CORBA是较小的内存需求和面向对象的环境（ORB库是C++库）。

2）IIOP引擎是更小的内存需求，非面向对象的环境（IIOP引擎是C库），建立在更底层TCP/IP网络通信的基础上接收和发送IIOP消息（如图2所示）。

IIOP引擎主要具有以下特性：

1）完全独立存在。IIOP引擎作为一个单独的C函数库实现，其API通常包括IOR（Interoperable Object Reference）、GIOG（General Inter-ORB Protocol）和CDR（Common Data Representation）等三个部分，允许应用程序使用该函数库同其它ORB通信，除了需要底层网络的支持外，不需要其它额外的支持。

2）较小的内存需求，IIOP引擎代码尺寸可以小到15K左右，而ORB库全部实现通常是150K左右。

3）确定的内存使用，支持端系统的强实时性。I-IOP引擎库不动态分配或释放内存，相反，当它需要获得或释放内存时，直接调用应用程序代码，由应用程序调用RTOS系统调用完成。

4）支持QoS。IIOP引擎实现的IIOP消息，允许端到端的通信过程中，传递QoS参数。

5）高效通信。IIOP引擎使用了几个程序优化技术，如在网络通信层间实现数据的零复制，以获得高的通信性能。

IIOP引擎的实现通常包含以下四个部分：

1）CDR封装引擎。它直接对简单的IDL数据类型进行解码，而对复杂的IDL数据类型，使用一个类型码（TypeCode）解释器进行解码。

2）类型码解释器。通过使用标准的IDL类型码，类型码解释器能够处理所有合法的IDL数据类型。

3）引擎框架。包含一个ORB的部分实现，该ORB实现的编程接口在CORBA2.0规范中有明确规定，一般情况下，它符合IDL C++映射。

4）IIOP相关模块负责同引擎框架通信，而CDR引擎负责发送、接收和调度IIOP消息。IIOP引擎主要实现嵌入式ORB与其它ORB之间的通信，将CORBA2.0 GIOP规范映射为TCP/IP，和普通CORBA客户/服务器通信相比，它在一个较低的层次上进行网络通信。对于一些资源非常有限的嵌入式系统，如没有足够的内存空间装载ORB函数库，或者装载面向对象的

应用程序，可以使用 IIOP 引擎代替 ORB，在其上用 C 语言开发嵌入式应用程序，实现互操作功能。

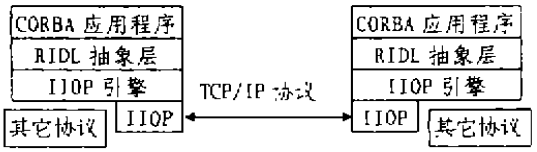

图 2 IIOP 引擎

4.3 嵌入式 ORB 采用一种专用通信协议

CORBA 模型目前采用的网络协议主要是 TCP/IP，但在嵌入式 ORB 实现中，可以根据应用需要采用其它专用的网络协议，如串行连接，在串行连接上用 SLIP/PPP 传送 TCP/IP 包，由于仅传输原始数据，需较小的处理能力就能够满足要求，同时，由于提供了一条点对点的专用通信链路，实时性也易于保证；或者提供 ATM、CAN 等网络存取能力，由于这些网络提供端到端的 QoS 保证，对于建立在有 QoS 保证网络基础上的嵌入式 ORB，也易于为应用程序提供端到端的 QoS 保证。为了满足嵌入式 ORB 的强实时要求，许多 ORB 产品（如 TAO）已经开始提供这些可选的协议（pluggable protocols）（包括 SLIP/PPP、UDP、ATM 等）。总之，CORBA 规范对 GIOP 协议，其底层通信协议并未作出明确规定，对于不同的 ORB 产品，可以根据应用需要实现 GIOP 底层的网络通信协议。另外，CORBA 规范提供了许多机制，鼓励在不同的环境间建立 ORB 网关，进行协议转换（如图 3 所示）。

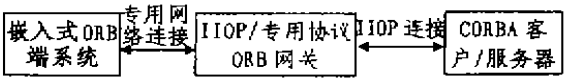

图 3 ORB 网关

结束语 嵌入式系统和 CORBA 模型结合成为开放的分布嵌入系统（OpenDistributed Embedded System），使嵌入式系统从专用系统走向开放系统，成为 CORBA-enable 设备，充分利用高度发达的、技术上健壮的分布式对象技术，在嵌入式系统和其它 C/S 计算系统之间实现无缝连接，更好地满足嵌入式系统对互操作等分布式处理的需求。

本文首先提出了嵌入式 CORBA 模型，接着分析了实时 CORBA 的局限性，最后重点讨论了嵌入式 CORBA 模型的结构及其三种主要实现方式。这三种实现方式，面向嵌入式系统，尤其是硬实时的嵌入式系统，能满足嵌入式系统的实时性和嵌入性要求，但仍有不足之处，如嵌入式系统通常是高可靠的系统，需要极强的容错能力，嵌入式 CORBA 缺乏容错的机制及策略。目前容错 CORBA 规范[9]已经推出，将成为 CORBA3.0 规范的一个组成部分，扩展嵌入式 CORBA 的容错能力，是下一步重要的研究工作。

参 考 文 献

1 徐仑峰，熊光泽，刘锦德 超微内核嵌入式实时操作系统的设计．计算机科学，1998，25(3)

2 骆志刚，刘锦德．实时 CORBA．计算机科学，1999，26(11)

3 Interoperable Objects for Distrubuted Real-Time Systems，ESP March'97

4 Forman G，Zahorhan J. The Challenges of Mobile Computering. IEEE Computer，1994，27(4)：38～47

5 Object Management Group. Minimum CORBA-Joint Revised Submission，OMG Document orbos/98-08-04 ed.，August 1998

6 Object Management Group. Real-Time CORBA-Joint Revised Submission，OMG Document orbos/99-02-12 ed. March 1999

7 Object Management Group. CORBA Messaging Specifications，OMG Document orbos/98-05-05 ed.，May 1998

8 Object Management Group Dynamic Scheduling，OMG Document orbos/99-03-32 ed.，March 1999

9 Object Management Group. Fault Tolerant CORBA Joint Revised Submission，OMG Document orbos/99-12-08 ed.，December 1999

10 Schmidt D C Techiques for Optimizing CORBA Middleware for Distributed Embedded Systems. In：Proc. of IEEE INFOCOM'99，New York，March 1999. 21～25

11 Schmidt D C. Measuring OS Support for Real-time CORBA ORBs In：4thIEEE International Workshop on Object-oriented Real-time Dependable Systems (WORDS'99)，Santa Barbara，California，January 1999 27～29