第14卷 第6期 航 空 学 报 Vol.14 No.6
1993年 6月 ACTA AERONAUTICA ET ASTRONAUTICA SINICA June 1993

分布实时计算远程操作引用机制研究

刘 熙 韩兆轩 康继昌
(西北工业大学 1102 教研室，西安，710072)

THE STUDY OF REMOTE OPERATION INVOCATION MECHANISMS FOR DISTRIBUTED REAL-TIME COMPUTING

Liu Xi, Han Zhao-xuan, Kang Ji-chang
(*Faculty* 1102, *Northwestern Polytechnical University*, *Xi'an*, 710072)

摘 要 传统的面向对象程序设计(OOP)机制由于缺少对面向对象分布实时计算(DRTC)特殊性的计算语义支持，不能很好地支持 DRTC 应用。解决上述问题的一个有效途径是扩展传统的 OOP 机制，为 DRTC 特殊性提供计算语义支持。提出了一个面向对象分布实时计算模型；研究设计了支持该模型的几种实时远程操作引用机制：同步远程操作引用、异步远程操作引用和多复本实时操作引用；并在一个实验分布实时计算系统 EDRTCS 中进行了实现。

关键词 实时远程操作引用，多复本分布实时计算，分割截止期协议，面向对象程序设计

Abstract Extending object-oriented programming(OOP) to distributed real-time computing (DRTC) environment has been and is the focus of researches around the world in recent years. In traditional object-oriented computing mechanisms, little computing semantics support is provided for distributed real-time computing applications. It is argued in this paper that the key to solve the problem is to integrate the distributed real-time computing semantics into the object-oriented computing mechanisms. Based on the above policy, this paper presents an object-oriented distributed real-time computing model. In support of the computing model, several real-time remote operation invocation (RT-ROPI) mechanisms are designed, namely, synchronous RT-ROPI mechanism, asynchronous RT-RORT mechanism and replicated real-time operation invocation (R-RTOPI) mechanism. All these RT-ROPI mechanisms are implemented in an experimental distributed real-time computing system --EDRTCS.

Key words real-time remote operation invocation, replicated distributed real-time computing, split deadline protocol, object-oriented programming

分布实时计算系统(DRTCS)由于与应用环境十分匹配并且具有潜在的优良性能，而得到广泛研究和应用[1,2]。DRTCS 的一个基本问题是分布实时计算(程序设计)环境研究，所要解决的是

(1)如何有效地支持分布实时计算软件开发；

(2)为分布实时计算(DRTC)应用的特殊性(即实时性、分布性和高可靠性)提供相应的计算语义支持

面向对象程序设计(OOP)可以有效地支持软件工程所要求的信息隐藏、数据抽象、模块化、代码共享(软件重用)等特征，是 DRTC 软件开发的有效途径。但是，传统的

1990 年 7 月 16 日收到，1991 年 9 月 15 日收到修改稿
国家教委博士点基金和西北工业大学青年科学基金资助课题

面向对象计算机制缺少对 DRTC 应用特殊性的计算语义支持，不能很好地满足应用的需要[3]。解决上述问题的一个有效途径是将传统的 OOP 机制与 DRTC 所要求的计算语义支持结合于一体，研究支持 DRTC 应用的面向对象计算机制。

1　计算模型

DRTCS 是指与环境的交互有着严格时间限制的分布信息处理系统。从信息处理的角度来看，DRTCS 与环境的交互可以抽象为对外部环境事件的监视和处理，在系统实现上即表现为完成一定功能的任务。一般地，任务总是由相应的分布实时计算程序(DRTCP)来实现；因此，怎样构造 DRTCP 并对其提供运行支持则是问题的关键。

1.1　面向对象分布实时计算程序构造

EDRTCS 是一个同构型 DRTC 实验环境，其硬件由 4 台 IBM-PC 计算机和自行开发的 MIL-STD-1553B 总线通信子网 1553BCN 构成。EDRTCS 支持称作实时对象(RTobj)、分布实时计算线(DRTthrd)以及实时远程操作引用(RT-ROPI)的程序设计抽象并在操作系统级向用户提供相应的 OOP 接口。

EDRTCS 中，应用软件分以下两部分管理

$DRTCP = <DRTthrd,\ PTO>$

$RTO = \{RTobj_i,\ i=1,\ 2,\ \cdots,\ m\}$

应用程序 DRTCP 通过 DRTthrd 引用一组 RTobj 上的操作完成其功能。RTobj 是传统对象机制的扩展，它遵从抽象数据类型(ADT)[4]的一般定义并支持时间封装(time encapsulation)思想[1]，为 DRTC 提供实时计算语义支持。DRTthrd 是描述计算的系统控制／调度抽象，它通过 RT-ROPI 可以在远程 RTobj 上移动[1,2]或分支[3]，从而实现分布计算；DRTthrd 还通过其实时计算特性 RTCC 来表征应用的实时性并通过 RT-ROPI 机制实现其依据于此的实时调度。

1.2　时间封装与分割截止期计算协议

一般地，应用的实时性要求由应用软件来表征并通过其被及时(实时)调度执行来满足。传统的 DRTCS 中，应用软件的实时性要求是以任务，即整个 DRTCP 为单位来描述的，例如任务的完成截止期。这种描述是粗糙的，可能造成系统行为时间上的不正确性。因为一个任务往往完成不止一个功能，而且功能与定时密切相关，即各功能模块有相对于同一环境事件的各不相同的完成截止时间；如果某个功能模块(而不是整个程序)不能在规定的时间期限内完成计算，就可能造成系统行为的不正确。

解决上述问题的传统方法是将程序的各个代码段与时间事件联系起来，以便能够预测系统的实时行为。带来的问题是：当系统需求变化而修改程序时，往往出现被修改的模块能满足其计算截止时间要求，而其它未被修改的模块却不再满足其计算截止时间要求。原因在于各模块不支持时间封装，即不能将各模块的计算时间错误限制在单个模块范围，而导致单个模块的计算时间错误超越模块边界。

因而提出一种分割截止期计算协议(SDP)，以支持 RTobj 操作引用的时间封装。SDP 的基本思想是将 DRTCP 的各功能模块，即 RTobj 操作与其时间要求结合起来描述应用软件的实时计算特性(RTCC)，并由 RT-ROPI 机制提供依据于此的调度运行支持(见图

1)。

设 D_{opr} 为某 DRTthrd 所引用 RTobj 操作的集合(D_{opr} 称作 DRTthrd 的引用操作域)，即 $D_{opr}=\{op_1, op_2, \cdots, op_m; m\geqslant 1\}$，则该 DRTthrd 的 RTCC 定义如下

$$RTCC=<C, D, P, d, \text{copd}>$$

$$C=<c_1, c_2, \cdots, c_m>$$

$$D=<d_1, d_2, \cdots, d_m>$$

$$P=<p_1, p_2, \cdots, p_m>$$

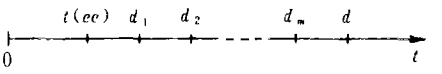

图 1 SDP 协议

其中，c_i 表示操作 op_i 所需计算时间；d_i 为 op_i 的计算截止时间；p_i 为 op_i 的引用周期；d 为整个程序的计算截止时间；copd 为当前引用操作的截止时间，是 DRTthrd 调度的依据，随当前被引用操作不同而改变。需要说明的是，RTobj 操作引用实际上是将其映射入引用 DRTthrd 空间，随 DRTthrd 的调度而执行。因此，每一次操作引用都必须修改引用 DRTthrd 的 copd。

2 实时远程操作引用及其实现

传统远程操作引用一般只要求给出被引用操作名、该操作所属对象名以及引用具体操作的引用参数，没有显式描述被引用操作时间要求的参数和依据于此的实时调度；而且缺少计算流(DRTthrd)远程移动或分支时对 RTCC 的远程继承支持，因而不能很好地支持分布实时计算。

EDRTCS 的 RT-ROPI 机制旨在弥补以上不足；它支持 SDP 协议并实现了计算流跨结点时被引用操作对引用 DRTthrd 实时计算特性的远程继承。

2.1 同步远程操作引用

传统操作引用是同步性质的，即引用对象在发出引用请求后被挂起，直到收到被引用操作的返回结果信息才继续执行，它支持计算流在对象间的移动。远程操作引用也应实现这种同步性质，以支持 DRTthrd 的远程移动。EDRTCS 中，这种同步性远程操作引用由 SynInvoke( )原语实现。SynInvoke( )原语格式如下

Val=SynInvoke(obj-name, op-name, [thrd-dsc], op-ddline, (opr-paras))

其中，obj-name 为被引用 RTobj 名；op-name 是被引用操作名；op-ddline 是被引用操作的完成截止时间；(opr-paras)为引用具体操作的具体引用参数和结果参数；thrd-dsc 是一个系统自动实现的隐式参数，用于实现 DRTthrd 远程移动时的语义一致性。参数 thrd-dsc 的定义如下

$$\text{thrd-dsc}=<\text{thrd-ID}, \text{RTCC}, \text{CPB}>$$

这里，thrd-ID 用于实现 DRTthrd 远程移动后的全局一致性寻址；RTCC 用于实现 DRTthrd 远程移动后实时计算特性的不变性；CPB 则用于维护 DRTthrd 远程移动后其操作引用权限的语义不变性。

2.2 异步远程操作引用

分布实时计算中，一个任务的执行常需要远程操作的并行计算；因此要求系统提供 DRTthrd 远程分支的并行计算机制。

同步远程操作引用仅支持 DRTthrd 的远程移动，不支持其远程分支；不能满足上述应用要求。EDRTCS 中提供一种异步远程操作引用机制来支持 DRTthrd 的远程分支并行计算，它由 AsynInvoke( )原语和 Wait( )原语所体现。AsynInvoke( )原语的执行将立即返回一个引用标识 IID；引用者则利用 IID 并通过显式地调用同步原语 Wait( )来同步分支 DRTthrd 和接收结果信息。

AsynInvoke( )原语与 Wait( )原语格式如下

Val = AsynInvoke(obj−name, op−name, [thrd−dsc], IID, (opr−paras));

Val = Wait(IID);

其中，thrd−dsc 也是系统自动给出的隐式参数，用于实现分支 DRTthrd 对原 DRTthrd 的语义继承；IID 为引用标识，它实际是操作引用序号 ISN，用于有选择地同步各分支 DRTthrd；其余参数同 SynInvoke( )原语。

2.3 实时计算特性 RTCC 的远程继承

EDRTCS 中，一个任务的执行表现为一个根 DRTthrd 的移动和分支，任务的实时性要求由 DRTthrd 的 RTCC 表征并通过 DRTthrd 的及时调度执行来实现。

由于 DRTthrd 是通过引用定义在各 RTobj 上的操作来实现其功能的，而 RTobj 操作可以是公共引用的，因此，被引用操作的调度特性应当随引用 DRTthrd 的实时计算特性的不同而动态改变，以适应不同任务的实时性要求。EDRTCS 中，操作引用实际上是将被引用操作映象入 DRTthrd 运行空间随其调度而执行(这保证了局部操作引用的 RTCC 继承)，因此，当 DRTthrd 同步或异步远程操作引用时，必须在远程结点上创建与引用 DRTthrd 一致的远程移动或分支 DRTthrd，以保证其 RTCC 语义不变性。EDRTCS 中，通过将引用 DRTthrd 的 RTCC 作为远程操作引用的参数来传递(见 thrd−dsc)，实现了对用户透明的 RTCC 远程继承。

2.4 被引用操作调度

传统操作引用一般不引起一次新的调度，因为操作引用不引起应用软件调度特性的改变。EDRTCS 中，由于 SDP 协议，每一次 RT−ROPI 均引起 DRTthrd 调度特性(copd)发生变化，因此要引起一次新的调度，增加了系统中的调度次数(开销)。但是，由于 DRTthrd 是一种“light−weight process”，其调度时环境切换(context switching)的开销不大，只要定义在 RTobj 上的操作粒度大一些，不会对系统的效率有大的影响；同时，这也与 OOP 是一种系统构造方法(programming−in−the−large)是一致的。

2.5 参数传递

分布环境下，以传指针方式传递参数实际上只是传递一个远程指针[4]，对参数的访问仍要通过通信网络，在效率上传值比传指针更为有效。且远程指针导致了结点间存储管理的交互增加，结点自治性降低。而 OOP 要求对象具有良好的封闭性，对象内部的处理过程对外部透明，不应该直接访问对象外部的变量。因此，EDRTCS 中，RT−ROPI 的参数传递采用传值方式进行。

EDRTCS 是一个同构型系统，各结点都采用相同语言(C 或 C++)进行程序设计，所以 RT−ROPI 参数的值传递采用整块复制(block copy)方式进行，而无需对参数的数据结构(类型)进行编码传送[4]。EDRTCS 假定，RT−ROPI 的参数类型检查在编译时进行，参数传递过程不进行类型匹配检查。

2.6 通信

同步和异步 RT-ROPI 的根本差别在于系统控制流的不同；而它们在通信上都需要传递两个消息：操作引用消息(InvkMsg)和返回结果消息(RetMsg)。EDRTCS 中，提供一种由 InvkMsg 和 RetMsg 构成的消息对协议(PMCP)来实现 RT-ROPI 通信。PMCP 消息格式如图 2 所示

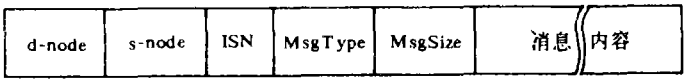

图 2 PMCP 消息格式

图中，d-node 为目的结点地址；s-node 为源结点地址；ISN 是远程操作引用序号，用以区别不同的消息对；MsgSize 表示消息长度；MsgType 为消息类型，分为以下几种：同步引用 SI，同步返回 SR，异步引用 AI，异步返回 AR，多复本引用 RI 和多复本返回 RR。

PMCP 建立在 1553BCN 数据报(1553B 消息)服务之上，而不是建立在虚电路通信服务之上。这是因为，InvkMsg 和 RetMsg 一般都很短，仅需一个或几个数据报即可实现 PMCP 消息传输，而不必建立虚电路服务，以减小通信开销。

3 多复本实时操作引用(R-RPOPI)

DRTCS 一般还要求极高的可靠性。由于实时控制过程的物理不可恢复性和实时性，一般采用冗余并行计算技术以提供容错实时计算能力[5]。传统的冗余并行计算是以任务(程序)为单位进行的。DRTC 的复杂性使得这种程序级冗余并行计算不再现实有效。首先，程序复杂性增大使得程序的模块化更加重要和流行；而完成不同功能的程序模块往往具有不同的可靠性要求，有的要求高一些，从而成为系统的可靠性瓶颈(reliability bottlenech)[5]。如果不加以区别，将程序的所有模块以机同冗余度并行计算，势必造成系统资源的浪费或影响程序可靠性。另外，DRTCP 往往是跨结点的，这也使得程序级冗余计算不再现实有效。解决上述问题的有效途径是在程序模块级实现不同冗余度的多复本并行计算。EDRTCS 中，提出一种多复本实时的对象计算模型，将实现关键任务的程序的全部或一部分以 RTobj 为单位冗余分布到多个结点并行计算，从而提供容错实时计算能力。该模型由多复本实时对象(R-RTobj)和多复本实时操作引用(R-RTOPI)两种计算机制体现。其中，R-RTobj 是指由分布于多个结点的同一 RTobj 的多个复本构成的独立计算实体，其定义如下

$$R\text{-}RTobj = <RTobj,\ NS>$$

$$NS = \{Node_i,\ i = 1,\ 2,\ \cdots,\ m\}$$

这里，NS 表示 R-RTobj 的复本成员所在结点集合。

3.1 多复本实时操作引用(R-RTOPI)

R-RTOPI 是指以下几种操作引用：RTobj 对 R-RTobj 的一对多(OM)操作引用；R-RTobj 对 RTobj 的多对一(MO)操作引用；R-RTobj 对 R-RTobj 多对多(MM)操作引用。

3.1.1 OM 操作引用 引用 RTobj 向被引用 R-RTobj 的每一个复本发引用消息；被

引用的每个复本都执行且仅执行被引用操作一次并向引用 RTobj 回送结果消息；引用 RTobj 接收并处理所有结果消息。

3.1.2 MO 操作引用　引用 R-RTobj 的每一复本向被引用 RTobj 发相同的操作引用消息；被引用 RTobj 接收多个相同的引用消息并回答多个相同的结果消息，但仅执行被引用操作一次。

3.1.3 MM 操作引用　引用 R-RTobj 的每一个复本向被引用 R-RTobj 的每一个复本发相同的操作引用消息；被引用 R-RTobj 的每一个复本接收多个相同的引用消息并回答多个相同的结果消息，但仅执行被引用操作一次；引用 R-RTobj 的每个复本接收并处理来自被引用 R-RTobj 的多个结果消息。

3.2 R-RTOPI 通信控制

由于 MM 操作引用中，对每一个引用复本，实际上是一 OM 操作引用；而对每一个被引用复本，实际上是一 MO 操作引用。因此，R-RTOPI 的通信控制由以下两算法实现:

3.2.1 OM 引用通信控制算法

(1)向被引用 R-RTobj 的每一个复本发送操作引用消息 InvkMsg。

(2)接收并处理被引用 R-RTobj 所有复本的返回结果消息 RetMsg。可采用不同的算法实现容错计算，如完全一致法(unanimous)，多数表决法(majority voting)。EDRTCS 采用表决法。

(3)若对被引用 R-RTobj 的某个复本的操作引用不成功(如超时)，则发流产消息 AbortMsg，通知远程结点流产该复本的操作。

OM 操作引用控制算法体现在 R-ROPI 原语(同 RT-ROPI 原语)中。

3.2.2 MO 引用通信控制算法

(1)接收 PMCP 消息，根据 ISN 来区别是否为同一次多复本操作引用消息或流产消息。

(2)若被请求引用操作未执行过，则调度执行之并保存结果信息；反之，不再执行该操作。

(3)发 RetMsg，返回结果消息。

MO 引用通信控制算法体现在通信管理进程中，由通信子网消息接收中断所激活运行。

参 考 文 献

1 Tokuda H, Mercer C W. ARTS: A Distributed Real-Time Kernel. ACM Operating System Review, 1989, 23(3):29～53

2 Northcutt J D. Mechnaisms for Realiable Distributed Real-Time Operating Systems: The Alpha Kernel. Florida, USA; Academic Press, 1987, 82～92

3 Tripathi A R. An Overview of the Nexus Distributed Operating System Design. IEEE Trans. on Software Engineering, 1989, 15(6):686～695

4 Herlihy M, Liskov B. A Value Transmission Method for Abstract Data Types. ACM Trans on Programming Languages and Systems, 1982, 4(4):527～551

5 Cooper E C. Replicated Procedure Call. ACM Operating Systems Review, 1986, 20(1):44～56