开放分布式对象系统中的通信智能体*

李巧云 李健东 康 卓 戴大为 康立山

(武汉大学计算机科学系 软件工程国家重点实验室 武汉 430072)

摘要 本文提出了一种新的通信智能体,称为通用对象请求智能体,通过超时技术提高开放分布式对象系统的客户/服务器模型通信的可靠性.本文提出通信智能体模型的目的在于提高请求的可靠性.在这个模型中,应用由时间段事务处理支持的反馈机制,对服务器完成请求的过程有更大的控制能力,并且客户以超时操作作为避免死亡的最后手段,从而使系统整体性能更加可靠.

关键词 通信智能体,超时,时间段,可靠性,客户/服务器模型.

在开放分布式对象系统 ODOS 中,应用对象请求软件模块的服务,服务的实现定义为客户/服务器模型(Client/Server Model).但是,在服务器提供服务的可靠性方面所做的工作还很少.

现存的系统如 OMG CORBA[1],LRTCA[2]和 ISIS[3],支持客户服务器模型之间通信的分布式对象系统,但是在客户/服务器模型中没有考虑时间限制.如我们所知,用户不能忽视不同对象之间的通信在整个计算时间中所占的比重.而且,对象作为客户,由于无法知道服务对象的状态,如出错、忙、正常等,在请求其它对象的服务时,必须根据时间决定其下一步的操作.这里的意思是,时间是对象可以依靠的最后一个手段.[4]因此,一个系统应该提供的最基本的特征是支持通信可靠性的实时特性.术语"超时"在 Takashio 的 DRO[4]模型中有介绍,用于客户退出延迟的事务以避免死亡.

在本文中,我们为客户提供超时机制以退出延迟的事务,同时,使用时间段机制设计通用对象请求智能体,主动调节客户与服务器之间的通信.

1 CORA 的目的

CORA(common object request agent)的目的是在支持分布式系统中的对象共享中提高系统的可靠性.自 1989 年以来,OMG(对象管理小组)建立了一个标准定义,为供应者之

---

* 作者**李巧云**,女,1965 年生,讲师,主要研究领域为智能体系统,分布式对象系统,软件工程,自然语言处理.**李健东**,1963 年生,讲师,主要研究领域为分布式系统,面向对象的系统设计与分析,软件工程.**康卓**,1970 年生,助教,主要研究领域为分布式系统,计算机网络.**戴大为**,1937 年生,教授,主要研究领域为人工智能,自然语言处理,计算机安全与保密.**康立山**,1934 年生,教授,博士导师,主要研究领域为计算机科学理论,并行计算.

本文通讯联系人:康立山,武汉 430072,武汉大学软件工程国家重点实验室

本文 1996-03-11 收到修改稿

间和供应者与用户之间的合作提供统一的标准.[5]但是,在目前的版本定义中,从客户到服务器的请求没有可靠的保障机制,可能导致对象请求死亡或系统死亡.因此,提出CORA以解决这个问题.我们研究CORA的基本目标如下:

(1)将时间信息、超时封装到客户的请求中;

(2)为智能体通信设计一种新的方法以支持通信可靠性并扩展通信原语的语义,使CORA既满足定义又包括时间限制;

(3)建立状态表,记录对象的未完成信息,使CORA作为对象请求交换器可以动态跟踪服务器的操作.

2 ODOS中的CORA模型

可以预计,下一个10年中计算机环境将是广泛分布、无所不在、开放和不停变化的.[1,7]在这样一个复杂的环境中,软件的可靠性对应用来说是非常重要的.

本文的重点在CORA为客户提供对象和服务器对象间通信的可靠性上.CORA的主要特征是个体化的智能体作为对象请求交换器,主动调节对象应用间的通信.

2.1 CORA的体系结构

本节介绍以下方法:对象请求的统一管理、知识库的使用和决策机制以及将请求信息广播到一组相关对象.ODOS的基本体系结构包括具有许多虚对象的对象库.当对象需要请求其它对象的服务时称为客户,而为其它对象服务的对象称为服务器.服务器将完成客户的请求.但是,由于各种原因,应用的完成不能得到保证.特别是当客户仅请求一个服务器的服务时,如果该服务器正忙或出错,客户必须永久等待.CORA依靠时间限制为客户与一组相关对象间的通信提供了一种机制以提高可靠性.

CORA在客户与服务器的请求通信中作为交换器.对象根据相关信息如分布的位置和功能分为虚对象组.当请求产生时,CORA首先根据请求的性质向一组对象广播请求,然后根据决策机制从回答的对象中选择一个对象作为服务器.最后,通知作为服务器的对象并同时建立状态表.在客户等待服务时,CORA管理服务器执行的过程.当客户超时仍然没有得到提交信息,将退出服务事务.如图1所示,CORA包含1个知识库(KB)、1个决策机制(DM)、1个临时状态表(ST)和1个反馈检查器(FV).它们的功能如下:

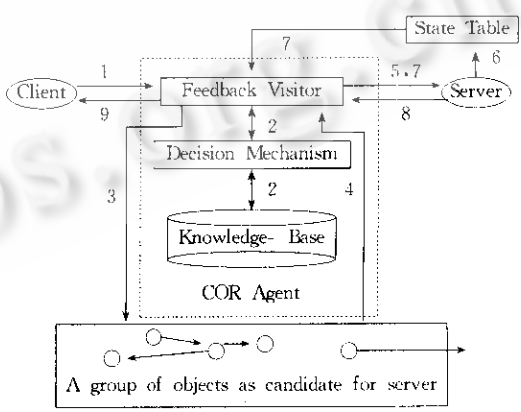


图1 CORA的体系结构

(1)KB:包含对象的分布位置、功能和服务历史.

(2)DM:具有逻辑结构的基于规则的机制,称为产生式——产生式的左边,即前件,声明一些需满足的条件;产生式的右边,即后件,声明要做的操作.

(3)ST:包括每个时间段服务器正在完成的服务的状态.如果在某时间段内没有收到消

息,ST 将自动记下'abnormal'.

(4)FV:接收 DM 的决策、向候选服务的对象广播以及通知被选为服务器的对象,然后为该服务器建立一个状态表以跟踪其操作.它每隔一个时间段检查状态表,当发现不正常状态时,处理服务器的非正常操作.当服务完成时向客户提交成功信息.

图 1 是这个客户/服务器系统的实现流程.其中服务采用 9 个步骤.以下算法描述了 CORA 是怎样工作的.

步骤 1:客户向 CORA 请求服务,其参数包括传送的时间限制;

步骤 2:CORA 收集参数并通过 KB 和 DM 基于参数选择一组虚对象作为候选服务器;

步骤 3:CORA 向这些对象广播;

步骤 4:对象应做回答;

步骤 5:CORA 根据回答选择一个对象作为服务器并将所有参数装入消息.消息建立以后,被传送到服务器;

步骤 6:广播和回答只是局部调用,这是因为虚对象存储器为 CORA 局部拥有.当消息到达服务器,服务器为该服务建立一个状态表并开始完成服务;

步骤 7:在服务期间,CORA 将管理状态表;

步骤 8:当服务器完成了服务就进行提交.此时 CORA 将通知客户服务已经完成,否则转步骤 9;

步骤 9:当期限到达时,客户将发出超时信息并且 CORA 的 FV 使服务器的事务退出.

2.2　通信协议

在 CORA 模型中,通信协议基于发送/接收[9]传输异步消息,并为对象的请求指定请求语义.协议通过建立包含状态和操作的状态转换图定义.状态对应执行条件,操作表示原语的异步发送/接收消息.这些原语消息由通信中发生的事件组成.

在 CORA 中,由发送/接收构成了 9 个消息原语.消息对应于图 1 中的步骤,其语义见图 2.

原语消息	语义
请求(Request)	请求消息
回答(Reply)	回答请求的广播
拒绝(Reject)	拒绝完成
退出(Abort)	由于超时而退出
提交(Commit)	正确完成
超时(Time-Out)	在客户上提醒超时
时间段(Time-Segment)	每个时间段检查状态表
广播(Broadcast)	向某些对象广播请求
选择(Select)	从候选者中选择一个对象

图 2 消息的语义

2.2.1　消息语义

不同于传统系统,CORA 的调用用于检查服务器的崩溃.其结果影响消息原语的语义.如图 2 所示,发送原语与接收原语不同.

客户 A 在 T_0 时刻请求服务并在预先确定的超时内等待提交.CORA 选择 B 为服务器,

将包含时间信息的参数封装到消息中并将消息传送给实际对象 B. B 得到消息并在 T_1 时刻开始执行. CORA 将在每个时间段管理 B 的执行. 给出一个合理的通用时间段的计算公式并非简单的事，它依赖于系统规模、运行速度、网络协议等因素. 在我们的模型中，作为模型的第 1 步，我们采用如下公式计算时间段的值. 假设 $TimeDo$ 是 B 执行和提交的正确时间.

$$\text{时间段}=(\text{超时}+T_0-T_1+TimeDo)/(2*UrgentFactor);$$

其中 $UrgentFactor>1$

超时$+T_0-T_1$ 是客户等待的期间，$UrgentFactor$ 是由 CORA 定义的请求紧急程度. $UrgentFactor$越大(即任务越紧急)，在超时内 CORA 检查状态表的次数越多. CORA 每隔一个时间段管理服务器的执行并在客户和服务器之间交换信息.

2.2.2　2 段提交协议

本节将看到由于服务器崩溃而引起的问题. CORA 系统的正常执行见图 3(a). 考虑在服务器执行完请求后发送提交之前崩溃. CORA 可以有 2 种解决方法. 令 CORA 在时间 $TimeClash$ 了解到服务器的崩溃.

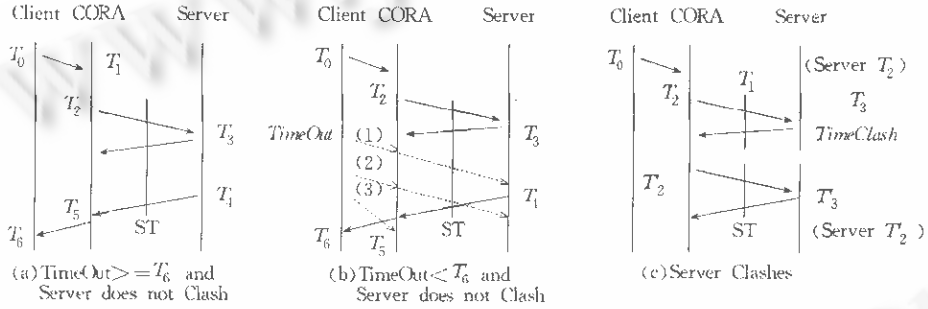


图3　2段提交协议

(1) 如果(超时—$TimeCrash$) 大于($TimeDo$+传送时间)，则选择另一对象为新的服务器；

(2) 如果 $TimeCrash$ 与超时接近，则悬挂直到客户超时并在知识库中记录异常；

(3) 如果非(1)或(2)，则向客户报告失败(退出)并在知识库中记录异常.

2.2.3　失败假设

我们称作为服务器的对象 B 对作为客户的对象 A **可用**，仅当 A 组织的请求被 B 及时完成且 A 通过 CORA 得到 B 对该执行的提示. 因此，B 对 A 为不可用出于以下 3 个原因：

(1) CORA 未收到 A 的请求. 在这种情况下，出现了通信失败(详见文献[6,8]).

(2) CORA 正确地发送到 B，但是 A 由于超时而退出 B 的事务. 见图 3(b).

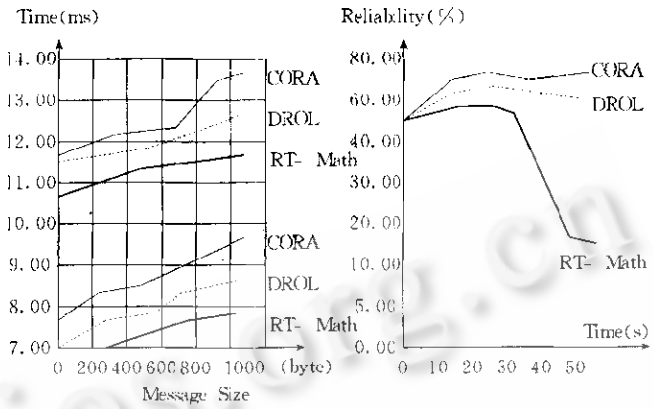
(3) 服务器 B 在服务完成以前崩溃. 这是图 3(c)所示的场地失败.

3　模型评价

在 RT-Mach4. 3BSD UNIX 操作系统上进行了模拟. 根据不同的对象数测量了超时方法的性能. 为评价 CORA，我们准备了由 C++写的对象集. 在对象集上做了 2 组试验. 一组对象通过 RT-Math 原语通信，另一组是通过 CORA 原语通信. 整个系统的可靠性和代价见

图 4. 通过模拟，可以得知下面的结果(测试时间单位为 ms)：

(1)智能体的引入，提高了系统的可靠性. 模拟中假设服务器的失败率与系统连续运行的时间成正比. 可以观测到，在 50s 后，服务器失败率上升到 80%，服务器的失败直接导致整个系统失败. 以同样的服务器失败率为数据模拟，引入通用对象请求智能体后，由于超时操作和服务器状态检查，能早期发现服务器的失败，避免整体系统死亡.



(1)通信代价　　(2)整体可靠性

图4

(2)通用对象请求智能体的引入增加了系统的时间开销，但反而提高了一定时间段内的效率. 一方面，由于时间段的计算和频繁的消息传送，系统时间开销增加约 12.549%. 另一方面，如果考虑系统死亡而带来的恢复时间的开销，引入 CORA 后，系统整体效率明显提高. 精确的测量依赖于系统恢复时间的长短和系统失败率. 在这方面将做更多的模拟.

4 结　论

本文的中心内容是在对象中加入超时和时间段原语以提高开放分布式对象系统中软件的效率和可靠性. 基于客户/服务器通信的标准对象的可靠性不够，人们对智能体的适应操作有一些研究.[9~12]本文集中讨论了由服务器的失败而引起的系统死亡. 我们相信，依靠这里提出的方法，使用新的消息原语可以显著提高智能体的性能.

(1)我们首先分析了在开放分布式对象系统中存在的问题，重点在通信的正确性和可靠性.

(2)第 1 次在请求服务中引入决策机制. 请求的实现是基于动态选择的一组对象而非一个对象.

(3)时间段方法的提出是为了使客户请求能被正确地完成. 实现过程可在每个时间段被检查，使失败的可能性降低.

参考文献

1　Hiroyuki SATO, Kunio OHNO. Comment on distributed object CORBA. Tokyo, Japan: Information Process, 1994, **35(9)**: 845~858.

2　Dugan J B, Lyu M R. System reliability analysis of a N-version programming application. NJ, USA: IEEE Trans. on Reliability, Dec. 1994, **43(4)**: 513~519.

3　Renesse R Van, Birman K P. Fault-tolerant programming using process groups. In: Brazier F, Johasen D eds., Proc. of Conference on Distributed Open Systems, Washington, USA: IEEE Computer Society Press, 1994. 135~152.

4　Takashio K. A study on an object-oriented distributed real-time programming language. Dissertation, Keio

University, Yokohama, Japan, April 1995.

5 Geoff Lewis. CORBA 2.0: Inter-operability-universal networked objects. TR: Document 95.3.xx, California, USA: Object Management Group, March 1995.

6 Bernstain P. Hadzilacos V, Goodman N. Concurrency control and recovery in database systems. Massachusetts, USA: Addison Wesley, 1987.

7 Genesereth Michael R, Ketchpel Steven P. Software agent. NJ, USA: Communication of ACM, Intelligent Agent. July 1994,37(7):48~53.

8 Tanenbaum Andrew S. Computer networks. New York, USA: Prentice-Hall, Inc., 1989.

9 Keith Decker, Victor Lesser. An approach to analyzing the need for meta-level communication. In: Proc. of IJCAI-93, France, 1993. 360~366.

10 Ichiro Satoh, Mario Tokoro. A timed calculus for distributed objects with o'clocks. In: Jeff McKenna eds., Proceedings of ACM Conference on Object-Oriented Programming Systems'93, NY USA: Spring-Verlag, July 1993. 226~245.

11 Silvano Maffeis. Adding group communication and fault-tolerance to CORBA. In: Proceedings of Conference on Object-Oriented Technologies, Canada: USENIX Association, June 1995. 135~146.

12 Birman K P, Renesse R Van. Fault-tolerant programming using process groups. In: Brazier F, Johasen D eds., Proc. of International Conf. on Distributed Open Systems, Washington, USA: IEEE Computer Society Press, 1994. 153~162.

COMMUNICATION AGENT IN OPEN DISTRIBUTED OBJECT SYSTEM

LI Qiaoyun LI Jiandong KANG Zhuo DAI Dawei KANG Lishan

(Department of Computer Science State Key Laboratory of Software Engineering Wuhan University Wuhan 430072)

Abstract This paper proposes a new communication agent, called CORA (common object request agent). The purpose of this approach is to improve the reliability of communication using time-out in open distributed system based on client/server model. In CORA model, the application client could have more control over server's processing the request through the feedback mechanism of timesegment transaction, and provide time-out's handling for preventing the bad propagation of the partial failure so that the whole system becomes more robust.

Key words Communication agent, time-out, timesegment, reliability, client/server model.