基于CORBA的分布对象容错机制研究与实现*)

The Research and Implementation of Fault-Tolerance Mechanism of Distributed Object Based on CORBA

李琪林 陈 宇 周明天

(电子科技大学计算机科学与工程学院 成都610054)

Abstract Presently, distributed object technology such as CORBA has increasingly become mature. More and more distributed application systems are implemented using the standard services and protocols provided by CORBA. The new-generation distributed systems such as real time systems, online paying systems and stock exchange systems demand assurance of dependability. Fault tolerance is a main way of assurance of system reliability. Thereby, it requires low-level CORBA infrastructure to provide fault-tolerance mechanism to ensure dependability and availability. This paper firstly discusses implementation strategy and system model of fault-tolerance CORBA object systems. Secondly, it describes main challenges and solutions during the design of fault-tolerance CORBA systems. Thirdly it introduces fault-tolerance CORBA prototype system-TBAFTS on top of a CORBA-compliant object middleware, TongBroker developed by us independently. Finally we give our conclusion.

Keywords Distributed systems, Software fault tolerance, CORBA, Fault-tolerance CORBA

一、引言

目前,以CORBA为代表的分布对象计算技术已日趋成熟,越来越多的分布式应用系统利用CORBA提供的标准服务和协议来实现[1]。基于CORBA的新一代的分布式系统,如分布式实时控制系统、在线支付系统和股票交易系统,需要可靠性保证。容错技术是分布式系统运行过程中可靠性保证的重要手段,可以在分布式系统的每一个层次实现,利用底层的CORBA基础设施提供容错机制具有显著的优势,既能够避免系统层为支持容错而做的巨大改变,又能够简化应用软件的设计。因此,容错CORBA已经成为国外CORBA研究的重点问题。冗余是实现容错的根本保证,可以分为硬件冗余、软件冗余和时间冗余。在面向对象系统中,软件冗余通常以多个冗余对象的方式存在,因此也可以称为实体冗余。实体冗余是容错CORBA的根本保证,确保系统在部分实体出现失效的情况下仍能够提供正确的服务。

当前,国外研究机构已就分布对象应用系统的可靠性和可恢复性进行了大量的研究工作,并提出了一些典型系统,如Electra[3],Orbix+Isis[4],Eternal[5],AquA[6]和DOORS[7],以及一些工具如Wolfpack[8],Watchd[9]和Firstwatch[10]。但这些系统和工具还存在一定的局限性并不能有效地解决分布对象系统的容错问题。例如Wolfpack,Watchd和Firstwatch基于进程来实现错误检测和恢复,因此无法有效地检测CORBA对象的失败,恢复对象间复杂的关系及有效的记录和恢复对象的状态。Electra和Orbix+Isis要求修改ORB来支持容错,系统的兼容性差。AquA和DOORS尽管系统兼容性好,并且提供了基于对象的检测和恢复机制,但没提供灵活的复制方式及底层设施控制的失败恢复。因此,如何在通用的CORBA平台上提供更加强大、高效的容错支持是当前研究的热点问题。

本文的第二节讨论了容错CORBA实现的系统策略。第三节描述了容错CORBA的系统模型,组成部分及各部分的相互关系。第四节详细分析了系统设计中面临的主要挑战及解决方案。第五节描述了在Windows NT上,基于我们自行研制的遵循CORBA规范的对象中间件平台TongBroker实现的容错CORBA原型系统TBAFTS(TongBroker-based Adaptive Fault-Tolerance System)。最后给出了结论。为了方便描述,下文将基于CORBA的分布对象容错简称为容错CORBA。

二、容错CORBA的系统策略和系统模型

2.1 容错CORBA的系统策略

如前所述,基于进程实现的CORBA容错具有明显的缺陷,不适于基于CORBA的分布对象系统。因此必须研究基于对象实现的容错,为分布对象应用系统提供可靠性和可用性。目前主要有三类方法实现容错CORBA[11]。

1)集成式策略 该方法要求修改ORB核心来支持容错,而ORB修改的程度取决于系统要求添加的功能。例如,可以在ORB中增加组成员一致性机制维护对象组的成员关系。该方法由于要求修改ORB,因此系统的兼容性差。采用该策略的典型系统如Orbix+Isis和Electra。

2)拦截器策略 基于该策略,客户请求在ORB外被拦截器捕获。拦截器修改客户请求,改变应用的行为或为应用增添新的功能。修改后的客户请求被映射到可靠的组通信系统中,送往服务器对象。采用该策略的典型系统如Electra和AquA。

3)CORBA服务策略 该方法将容错机制作为标准CORBA服务的一部分予以实现,因此不要求修改ORB,系统的兼容性好。客户请求该服务获取对象组的信息,从而调用该对象组的操作。容错服务负责管理和维护复制对象及相关对象的状态。采用该策略的典型系统如DOORS。

2.2 容错CORBA的系统模型

*)本文获四川省重点科技计划项目基金资助。李琪林 博士研究生,主要研究方向为分布式系统,对象中间件技术。陈 宇 博士研究生,主要研究方向为实时操作系统,软件容错技术。周明天 教授,博士生导师,主要研究方向为网络,对象中间件技术,应用服务器。

我们认为典型的容错CORBA系统模型应包括以下几部分:复制管理器,属性管理器,组对象工厂,对象组管理器,错误检测和通知器,本地对象工厂。其中属性管理器,组对象工厂和对象组管理器隶属复制管理器。其结构如图1所示。下面我们简要地描述一下基于CORBA的分布对象系统容错模型各部分的功能及相互关系。

复制管理器 是整个容错CORBA系统模型的核心。它负责管理复制对象,实现对象冗余,为应用提供容错支持。复制管理器又包括属性管理器,组对象工厂和对象组管理器。

(1)属性管理器。定义对象组的属性如复制策略,组成员关系,失败恢复机制和初始复制对象数或最大复制对象数。组成员关系和失败恢复机制可由应用或容错CORBA基础设施控制。

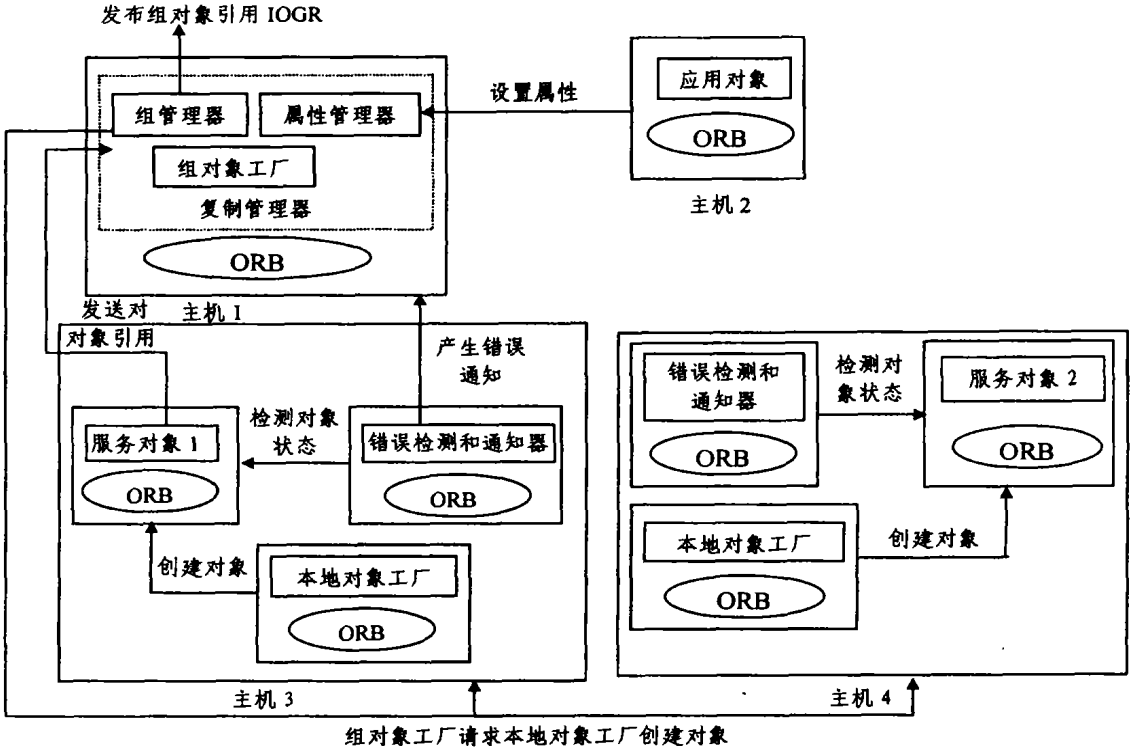

图1 基于CORBA的分布对象系统的容错模型

(2)组对象工厂。组对象工厂根据复制管理器的请求,创建对象组。在本地对象工厂的支持下,创建组对象成员的复制对象。另外,通过与属性管理器通讯,对象工厂维护对象组的初始复制对象数和最大复制对象数。

(3)对象组管理器。对象组管理器为应用或底层基础设施提供相应的接口,创建、增加、删除组成员,维护组成员的一致性。

错误检测和通知器 错误检测器通过相应机制检测错误的CORBA对象,并将错误报告发送到错误通知器。错误通知器再将错误信息传递给复制管理器。复制管理器根据错误信息,进行错误处理和恢复。

本地对象工厂 与组对象工厂通讯,负责处理组对象工厂提交的请求,创建复制对象。

三、系统设计面临的主要挑战和解决方案

3.1 挑战1:复制策略的选择

与其它的容错系统一样,对象冗余是基于CORBA的分布对象系统实现容错的关键,通过复制、错误检测和恢复,容错CORBA为分布对象应用提供支持,实现容错,保证系统可靠和可用。但是不同的应用和环境可能要求不同的冗余级别实现容错。为了有效地利用资源,在保证系统可靠性的前提下,提高资源的利用率,实现系统任务最大的吞吐量,我们采用三类复制策略:冷复制,暖复制和热复制,支持容错。冷复制仅包括一个主对象(primary replica),负责处理客户请求。一旦主对象失败,系统会启动一个新的对象来作为主对象,继续处理客户的请求。该方法相对简单,特别适于无状态服务器,但是系统开销大,性能差。暖复制包括一个主对象,一个或多个从对象。一旦主对象故障,系统会通过选举从从对象中提升一个作为新的主对象。主对象处理所有的客户请求,并负责与从对象状态的同步。基于该方法,从对象总在运行且与主对象同步,因此客户不必等待新对象的重启和恢复,系统开销小。热复制中所有的对象均是主对象,它们独立地处理客户请求,并就处理的结果达成一致并返回客户。该方法基本上没有开销,因此系统的性能好,但要求底层可靠的多播组通信协议提供支持。在具体应用中,不同的应用可以根据自身需要和系统资源的状况,通过属性管理器设置复制策略,选择不同复制方法,实现不同的冗余级别,保证系统最大的灵活性。

3.2 挑战2:失败恢复策略分析

失败恢复是所有容错系统必须考虑的问题。同样,容错CORBA也必须解决错误对象的恢复问题。最常用的恢复方法是日志方式,日志中记录系统最近发生的操作。一旦系统失败,根据日志信息,系统能重做或回滚相关操作,使系统恢复到失败前的状态。在容错CORBA系统中,我们也采用日志方式进行失败恢复。为了保证系统最大的灵活性,我们选择两类日志恢复策略:应用控制的失败恢复和基础设施控制的失败恢复。对于应用控制的失败恢复,应用自身负责记录日志和进行失败恢复;对于基础设施控制的失败恢复,我们采用拦截器拦截所有客户对象发送到服务器对象的IIOP消息并将拦截

的消息记入日志。一旦服务器对象失败，根据不同的复制策略（冷复制，暖复制或热复制），失败恢复机制使新的主对象恢复到服务器对象失败前的状态。

3.3 挑战3：对象组的定位

对象引用是分布对象技术的关键，有效地实现对象的定位和互操作[12]。容错 CORBA 的基础是对象冗余，一个分布对象包括一个或多个复制对象，该对象和它的复制对象构成对象组。因此，如何有效地定位对象组，将客户请求透明地传递给适当的对象成为分布对象容错的关键。为此，对象管理组织定义了 IOGR（Interoperable object group references）。IOGR 结构如图2所示。IOGR 包括多个 TAG—INTERNET—IOP 标记，定位服务器对象组。客户使用 IOGR 透明地传递请求，调用对象组上的操作，而无需了解服务器复制对象的存在。一旦服务器对象失败，客户可以使用 IOGR 中的对象引用访问复制对象，直到请求被复制对象处理。

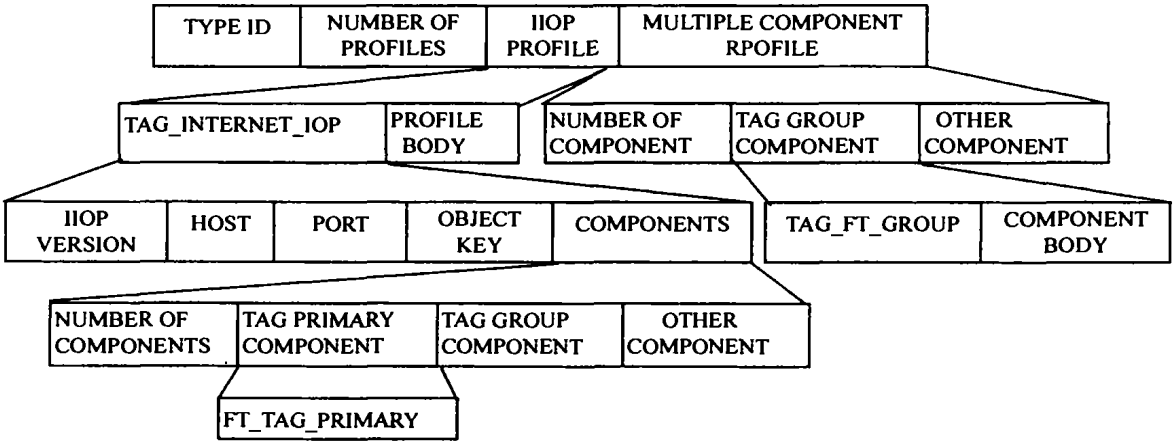

图2 IOGR 的结构

3.4 挑战4：错误的有效检测

正常情况下，服务对象接受客户请求，调用相应的操作并将结果返回客户。当服务对象失败时，系统根据复制策略（冷复制，暖复制和热复制），启动恢复机制使新的主对象恢复到系统失败前的状态，透明地实现容错。因此有效的错误检测也是实现容错 CORBA 必须解决的问题。为此，我们在容错 CORBA 系统中提供两类错误检测方法，生成错误报告。一种方法是 Pull 模式。该方式由专门的错误检测器定期地测试服务对象是否正常。一旦检测到服务对象失败，错误检测器产生错误报告并通知系统启动失败恢复机制；另一种方法是 Push 模式。该方式由服务对象定期地生成'心跳'信息，并发送给错误检测器。一旦错误检测器无法检测到'心跳'信息，便认为该服务对象出错，并通知系统启动失败恢复机制。

四、实现

基于前文讨论的容错 CORBA 系统模型及系统设计面临的主要挑战和解决方案，我们在 Windows NT 平台和我们自行研制的新一代面向对象的中间件 TongBroker 上实现了容错 CORBA 原型系统 TBAFTS（TongBroker-based Adaptive Fault-Tolerance System）。为了维持 ORB 核心不变，使系统具有良好的兼容性和扩展性，我们将容错机制作为 TongBroker 的服务实现。为了保证互操作性，系统将 IIOP 协议实现为基本的协议构件，而 GIOP 消息的打包和拆包采用符合 CDR 格式的流，它们被静态配置在核心中；另外，应用 API 和常规 ORB 基本一样，从而使应用在不同的 ORB 之间移植性更好。同时，TBAFTS 系统的所有部分均作为标准的 CORBA 对象实现，定义为 IDL 接口，通过 TongBroker 提供的映射机制，映射成 C++类。

正如前文讨论的一样，TBAFTS 系统也包括复制管理器，本地对象工厂和错误检测通知器。复制管理器又包括属性管理器，组对象工厂和对象组管理器。但本系统的设计策略更强调系统的灵活性，因此实现上与常规容错 CORBA 系统最大不同包括以下几点：

（1）*支持多种复制策略* 系统目前支持3种复制策略（冷复制，暖复制和热复制），允许不同应用和用户通过属性管理器设置复制策略，实现不同的冗余级别。因此能有效地利用资源，在保证系统可靠性的前提下，提高资源的利用率，实现系统任务最大的吞吐量，保证系统最大的灵活性。尤其热复制要求底层可靠的多播组通信协议提供支持，目前绝大多数系统均不能支持热复制。利用 TongBroker 提供的可插卸协议框架，ORB 核心可以静态或动态地增加对新协议的支持，因此能够方便地配置任何组通信协议。组通信协议被加载后插入到核心协议槽中，就可以起作用，这极大提高了系统的可扩展性和可伸缩性。

（2）*基础设施控制的失败恢复策略* 系统基于日志方式进行失败恢复。但现有的许多容错 CORBA 系统将失败恢复交给应用，这无疑加重了应用开发者的负担。为此，除了提供应用控制的失败恢复外，我们还利用 TongBroker 提供的拦截器在系统中实现了基础设施控制的失败恢复，从而简化应用的开发。

（3）*支持 PUSH 和 PULL 方式的错误检测* 为了提供灵活的错误检测，系统实现了 Pull 接口和 Push 接口。Pull 接口定期输出 is—alive()操作，而 push 接口用 heartbeat()操作，定期发送'心跳'信息到错误检测器。这极大丰富了错误检测方法，方便系统更准确地诊断错误。

结论 新一代关键业务系统具有内在的分布性，而且随着 CORBA 技术的日益成熟，越来越多的这类系统基于对象中间件技术构建。通常这类系统要求容错支持，保证系统的可靠和可用。因此很有必要研究容错 CORBA 技术，为分布对象系统提供容错机制。本文提出了容错 CORBA 系统模型并设计和实现了容错 CORBA 的原形系统 TBAFTS。为了实现不

（下转第185页）

4. 主负载信息表主机的迁移和崩溃策略

主负载信息表主机本身也进行计算任务，即也参与了负载的迁入与移出操作，故其本身的负载也很有可能超出其崩溃上限，特别是当此主负载信息表主机崩溃时为了保持系统的稳定与可靠性，有必要在此时进行主负载信息表主机的迁移和重选新择主负载信息表主机的操作。

4.1 主负载信息表主机迁移策略

主负载信息表主机的迁移策略如下所示：

if 主负载信息表主机负载超过上限 then

step1：调用图3的负载信息收集机制收集各主机当前最新负载信息；

step2：选择当前负载最轻的节点作为新的主负载信息表主机，其主负载信息表的映像即为当前的主负载信息表；

step3：对原主负载信息表主机调用自适应动态负载均衡算法，但此时不需要重新收集负载信息；

step4：if（原主负载信息表主机负载＜上限）then 恢复其为主负载信息表主机。

4.2 主负载信息表主机崩溃策略

主负载信息表主机崩溃策略如下所示：

step1：从任一主机负载信息表映射中选一当前负载最轻的节点作为主负载信息表主机；

step2：删除原主负载信息表主机，调用图3的负载信息收集机制；

step3：if 原主负载信息表主机恢复，则添加此主机，再次调用图3的负载信息收集机制。

结论 本文结合集中和分散以及全局和局部动态负载平衡算法的优点，提出了一个基于主负载信息表的动态负载平衡模型，并结合此模型，提出了一个有效的自适应动态负载平衡算法。从前面的分析中，我们可以看出此模型的优点有：

1. 可靠性，无论哪个结点崩溃，都不会影响其它节点的工作；

2. 稳定性，不会造成各节点不断探询负载信息而任务得不到执行的情况，也即有效地避免了任务的“抖动”；

3. 良好的可扩展性，在该模型中可以随时加入或删除主机，不仅适用于同构系统还可用于异构系统；

4. 高效性，每个处理机仅根据各自的负载信息表进行自适应性负载平衡决策，并且仅在负载超过上下限时才进行负载信息的被动收集，最大限度减少了动态负载平衡过程中的开销；

5. 通用性，可适应于不同类型的任务。

正是基于以上一些优良的特性，故该负载平衡模型要优于当前一些传统的模型。

参 考 文 献

1 刘红霞，李东，等. 工作站网络中负载参数的一种收集方法. 小型微型计算机系统，2000，21(3)：261～263

2 陈志刚，李登，曾志文. 分布式系统中动态负载均衡实现模型. 中南工业大学学报，2001，32(6)

3 鞠九滨，杨鲲，等. 使用资源利用率作为负载平衡系统的负载指标. 软件学报，1996，7(4)：238～243

4 李登，陈志刚. 分布式系统负载均衡策略研究：[中南大学硕士学位论文]. 2002

5 Svensson A. Dynamic Alternation between Load Sharing Algorithms. In：Proc of the 25th Hawaii Intl. Conf. on System Sciences，Hawaii，Jan. 1992，1：193～201

6 晏荣杰，张玉明. 多处理机系统的自适应动态负载平衡算法研究，计算机应用，2001，21(7)：34～36

7 肖侬，卢宇彤，等. 一个基于网络并行计算环境的动态负载分配算法. 计算机研究与发展，1999，36(2)：238～241

8 Chi-Chung Hui，Chanson S T. Theoretical Analysis of Heterogeneous dynamic load-balancing problem using a hydrodynamic approach. Journal of Parallel and Distributed Computing，1997，43(2)：139～146

9 Zaki M J，Li W，Parthasarathy S. Customized Dynamic Load Balancing for a Network of workstations. Journal of Parallel and Distributed Computing，1997，43(2)：156～162

（上接第173页）

同的冗余级别，我们引入了多种复制策略（冷，暖和热复制），并结合 TongBroker 提供的可插卸协议框架，实现对象冗余，增加系统的灵活性；为了有效地进行错误检测，我们实现了 Push 和 Pull 机制，方便系统更准确地诊断错误；为了保证系统从失败中恢复，我们不仅提供应用级的失败恢复而且提供了基础设施级的失败恢复，简化了应用的开发。实践证明，容错 CORBA 对于保证分布对象系统的可靠性和可用性是可行的，能够推动高可靠和高可用分布对象系统的进一步发展。

参 考 文 献

1 Vinoski S. CORBA：Integrating Diverse Applications Within Distributed Heterogeneous Enviroments. IEEE Communications Magazine，1997，14(2)

2 Cristian H. Understanding fault-tolerant distributed systems. Communications of the ACM，1991，34(2)：56～78

3 Maffeis S. Adding Group Communications and Fault-Tolerance to CORBA. In：Proc. of the Conf. on Object-Oriented Technologies，Monterery，CA，USENIX，1995

4 Birman K，van Renesse R. Reliable Distributed Computing with the Isis Toolkit，IEEE Computer Society Press，Los Alamitos，1994

5 Narasimhan P，Moser L E，Melliar-Smith P M. Using Interceptors to Enhance CORBA. IEEE Computer，1999，32(7)

6 Sabnis C，et al. Proteus：A Flexible Infrastructure to Implement Adaptive Fault-Tolerance in AquA. In：Proc. of the 7th IFIP IWC in DCCA，1999. 137～156

7 Natarajan B，et al. DOORS：Towards High-performance Fault Tolerance CORBA. In：Proc. of the 2nd Distributed Applications and Objects（DOA）conference，Antwerp，Belgium，Sep. 2000. 21～23

8 MSCS，Microsoft NT Server Edition. www.microsoft.com，1998

9 Huang Y，Kintala C. Software Implemented Fault Tolerance：Technologies and Experience. In：23rd Intl. Symposium on Fault-tolerance Computing（FTCS），Toulouse，France June 1993. 2～10

10 Veritas. Veritas FirstWatch. www.veritas.com/us/products/firstwatch，2000

11 Narasimhan P. Transparent Fault Tolerance for CORBA：[Ph. D. thesis]. University of California，Dept. of Electrical and Computer Engineering，Santa Barbara，CA，Dec. 1999，Available as：Technical Report UCSB 99-18

12 Object Management Group. The Common Object Request Broker：Architecture and Specification，2. 3 ed.，June 1999