维普资讯 http://www.cqvip.com

可信中间件——技术现状和发展

李琪林[1] 周明天[2]

(四川电力试验研究院系统室 成都 610072)[1] (电子科技大学计算机学院 成都 610054)[2]

摘 要 本文全面而系统地介绍了可信中间件系统的技术、研究工作及关键问题的现状。在一些基本概念介绍的基础上,按照研究层次、领域问题和实现方式等不同角度,对研究工作进行了深入的分析,并详细讨论了可信中间件系统的研究现状。随后,列举了一些有代表性的系统和相应的技术特点。最后指出了可信中间件系统未来的研究发展方向。

关键词 可信性,可信计算,可信中间件

Dependable Middleware: Survey and Further Evolution

LI Qi-lin[1] ZHOU Min-tian[2]

(Power System Department, Sichuan Electric Power Test and Research Institute, Chengdu 610072, China)[1]

(Computer Science and Technologies School, University of Electronic Science and Technology, Chengdu 610054, China)[2]

Abstract This paper introduces the state of the art of dependable middleware system comprehensively in terms of techniques, research works and key issues. Based on the brief introduction to some basic concepts, the paper goes into deep analysis on the relevant efforts from the different views on the research hierarchy, field problems and implementation framework. Some typical systems and their key features are investigated. Finally, the future direction of dependable middleware system is presented.

Keywords Dependability, Dependable computing, Dependable middleware

1 引言

上个世纪90年代以来,随着中间件的产生和广泛采用,中间件技术在分布应用中发挥着日益重要的作用。但日趋增多的应用领域,尤其是电力、电信、航空航天、工业实时控制等关键业务应用领域,对中间件系统提出了新的挑战。在这些关键业务应用领域,一旦系统发生故障,将会给社会造成难以估量的生命财产损失,甚至带来巨大的灾难。因此,如何为这类应用提供可信性支持,也就成为推动中间件技术发展的关键。

然而,当前多数中间件系统如CORBA[1],DCOM[2]或RMI[3]大都侧重于为分布应用提供通用的支持如消息通信、位置透明、语言透明和互操作等,缺乏支撑可信计算的关键机制如对象复制、一致性维护、故障检测和恢复等;在面临节点失效、对象崩溃、应用值故障和网络分区等问题时,也日益暴露其局限性。但从应用需求角度看,可信应用比普通应用更加复杂,应用开发者更加需要中间件平台的支持。因此,对可信中间件技术的研究也就成为中间件技术领域的重要课题。

可信中间件系统在受到广泛关注和研究的同时,也面临新的问题和挑战。本文全面而系统地介绍了可信中间件系统的技术、研究工作和发展现状,具体内容包括:基本概念、研究工作分类和现状、典型系统及未来发展方向等。

2 基本概念

2.1 可信性

一般来说,可信性(Dependability)被定义为计算机系统的这样一种性质,即它所提供的服务有理由认为是可以信赖的。系统提供的服务是用户可以感知的一种行为,而用户则是能与之互动的另一个系统(人或者物理的系统)。

可信性作为描述计算机性能的一种指标,是对计算机系统所交付服务性质的一种定性度量,用于表示系统可以提交足以信赖的期望服务的能力。按照应用对其属性的量化,可信性通常包括以下几方面的含义:可用性(Availability)、可靠性(Reliability)、保险性(Safety)、安全性(Security)[4-6]。相应地,可信性除了要求系统具有容错能力外,也着眼于系统容忍风险和抵御灾难的能力。因此,目前可信性更多用于评价开放、异构、分布的网络环境下计算机系统的整体性能。

2.2 可信计算

可信计算(Dependable Computing)的概念可以根据可信性做不同程度的定义。早期可信计算的研究主要集中在操作系统自身安全机制和硬件支撑环境。相应地,此时的可信计算被称为"可靠计算",与容错计算领域密切相关。随着Internet技术的发展,可信计算的含义不断地演进,由早期侧重于硬件可靠性和可用性到针对硬件平台、软件系统、服务的综合可信,适应了Internet应用系统不断发展的需要。

广义地讲,可信计算是一个涵盖计算机硬件(本地硬件、网络部件)、软件及服务可信性的综合概念,其研究与计算网络技术、容错技术、信息安全技术直接相关。具体而言,可信计算的研究内容包括:可信的软件工程技术、容错技术、容侵系统、安全分布式计算技术、无线网络安全技术、系统信任管理等等。

2.3 可信中间件

李琪林 博士,主要研究方向为电力自动化、分布对象技术、可信计算技术等;**周明天** 教授,博士生导师,主要研究方向为计算机网络、分布对象技术、并行分布处理等。

维普资讯 http://www.cqvip.com

现代分布式系统的一个普遍趋势是采用中间件作为应用开发和运行的平台。对可信应用而言，可信中间件(Dependable Middleware)具有重要的应用价值，因为对可信性的支持往往需要实现大量复杂的机制，如对象复制、一致性维护、故障检测和恢复等，而这些可以在中间件层次中实现。类似于常规中间件，可信中间件也可提供消息通信、远程方法调用、事务、组件等基础设施和服务。借助可信中间件，分布计算网络环境和可信性支撑机制的复杂特征和变化可以对应用透明，可信应用可以类似于常规应用进行开发。

3 研究工作分类

3.1 面向层次的分类

总体来说，可信中间件技术的研究大体上分为以下几个层次：(1)操作系统层，即在操作系统中提供支持可信计算的关键机制，满足应用的可信性要求，如 California 大学 Irvine 分校的 DREAM 内核[7]，Berlin 工业大学的 MARS[8]，Massachusetts 大学 Amherst 分校的 Spring[9] 等；(2)可信计算平台，即从中间件平台角度进行研究，支持对应用透明或部分相关的可信计算，研究工作如 Illinois 大学 Urbana-Champaign 分校的 AQuA[10]，California 大学 Santa-Barbara 分校的 Eternal[11]，Cornell 大学的 Maestro[12]，Lucent 贝尔实验室的 DOORS[13]，Carleton 的 Chameleon[14]，Zurich 大学的 Electra[15]，IONA 的 Orbix+Isis[16]，瑞士联邦工学院 Lausanne 分校的 OGS[17]，California 大学 Irvine 分校的 ROAFTS[18]，法国 LAAS-CNRS 的 FRIENDS [19]，Carnegie Mellon 大学的 MEAD[20] 等；(3)高级服务和设施，即在中间件平台之上针对可信应用的服务和设施，如 Zurich 大学的 Piranha[21]，Veritas 的 FirstWatch[22]，Lucent 贝尔实验室的 Watchd[23]，Microsoft 的 Wolfpack[24] 等。

3.2 面向研究问题的分类

按照研究的主要问题，可信中间件的研究包括：(1)从体系结构角度研究灵活的可信计算结构或模式，如法国 LAAS-CNRS 等研究机构合作开发的 Delta-4[25]，英国 York 大学等机构合作研制的 GUARDS[26]，Chameleon 等；(2)针对分布计算环境的高度动态性，研究中间件系统对环境的自适应容错能力，如 ROAFTS，AQuA，MEAD 等；(3)研究如何使用反省技术为分布应用提供可信性支持，这方面的研究工作如 Virginia 大学的 Legion[27]、法国 LAAS-CNRS 的 FRIENDS 等；(4)研究如何充分利用复制技术满足可信应用对高可用服务的需求，如 Active Replication[28]，Primary Backup[29]，ROWAA(Read Once/Write All Available)[30]，Majority[31]，Tree[32] 等；(5)扩展已有的分布对象中间件平台 CORBA 或构建崭新的中间件基础设施，满足分布应用对实时性、QoS、资源管理及可信性的综合要求，如 Lucent 贝尔实验室的 QuO[33]，Washington 大学的 TAO[34]，California 大学 Santa-Barbara 分校的 Realize[35]，加拿大多伦多大学的 Agilos[36]，California 大学 Irvine 分校的 Compose|Q[37] 等；(6)研究多 Agent 系统中的可信性保障机制，为应用提供自适应容错支持，如法国 LIP6 等研究机构开发的 DARX[38]，Oregon 大学的 AAA[39]，Chameleon 等。

3.3 面向实现方式的分类

针对现有的分布对象中间件计算平台 CORBA 缺乏可信计算支持机制，有相当一部分研究工作集中于如何为 ORB 及其应用增强容错能力。按照实现方式划分，这些研究工作大致分为以下三类：(1)集成法，即通过修改 ORB 核心来提供必要的可信性支持。该方法将管理对象组的方法嵌入到 ORB 核心中，允许系统中任一对象成为对象组成员，并能有效地区分单一对象引用和对象组引用，而且 ORB 修改的程度取决于系统要求扩展的功能。研究工作如 Electra，Orbix+Isis，Maestro。(2)拦截法，即在 ORB 之下拦截客户方请求，修改请求的参数以改变应用的行为或为应用增添可信性支持。更改后的客户调用被重新映射到底层可靠的组通信系统中，并发送到服务器对象。这方面的研究工作如 Eternal、AQuA。(3)服务法，即将可信性支撑机制和策略作为 ORB 之上的 CORBA 服务予以实现，因此该方法无需修改 ORB 核心。客户请求可信计算服务获取复制对象组的信息，从而调用复制组。可信计算服务负责对象复制、复制管理、故障检测及日志恢复等。研究工作如 英国 Newcastle 大学的 NewTop[40]，意大利 Rome 大学的 IRL[41]，以色列 Technion 工学院的 FTS[42]，DOORS，OGS 等。

4 可信中间件系统的关键问题

4.1 复制一致性

复制技术是可信计算典型而重要的研究分支，尽管这方面的研究工作并不是新近出现的，但仍有相当多的工作需要开展。复制一致性技术用于保证所有冗余副本之间数据的一致性。具体而言，复制一致性技术需要解决的问题包括：一致性定义、数据更新、更新冲突解决以及分区网络中一致性维护等等。

针对一致性定义问题，过去研究大都将一致性定义为：强一致性和弱一致性。强一致性规定所有副本必须同步更新，因此降低了数据的可用性和访问速度；弱一致性允许副本非同步更新，但这可能会导致冗余副本间出现暂时不一致。为了适应不同应用对一致性的不同需求，目前许多研究着眼于对一致性定义进行扩展，引入时间、概率条件和操作顺序等属性，以提供更加灵活、可控的一致性模型[33,43,44]。

针对数据更新，需要解决的问题包括：更新实施的位置；更新状态的传播；更新传播的方向；更新操作的排序等等。

更新实施的位置本质上是谁实施更新的问题。例如，在主副本系统[45]中，某一副本被指定为主副本，其它副本为从副本。所有的更新操作先被主副本接收并执行。执行完后，它再将更新消息发送给其它从副本，以同步它们的状态。对称副本系统则允许任意的副本发起更新请求，实施更新。副本间通过定期地交换更新信息，解决可能的更新冲突，保证所有副本的一致性。

为了传递更新状态，目前的研究工作主要采用三种方法：一种方法是将更新后的整个副本的状态传递给其他副本；另一种方法则是递增地传递副本的状态，即只传递更新过的状态给其它副本[46]；还有一种方法是仅传递更新操作本身，接收到更新操作的副本在本地实施更新，保证副本一致性。这类方法也称为 Log-Transfer。

更新传播方向实质是谁负责发起更新的问题。例如，某些系统采用 Pull 技术，系统中的每一副本轮询其他的副本，获取他们的更新信息，改变自身的状态。又如，一些系统使用 Push 技术，总是由执行更新操作的副本负责发起更新的传播。再如，在 Blind Push 方法中，当前执行更新操作的副本任意地选择一组或向所有的副本传播新的更新操作。最后，在一些 Gossip 或 Anti-Entropy 协议[46]中，副本以成对的方式，交换彼此的更新信息，保证状态一致性。

为了维护一致性，所有冗余副本除了必须接收到相同的更新操作集外，还必须以适当的顺序执行更新操作。针对这一问题，已有研究大致分为以下两类方法：全序和偏序。全序

维普资讯 http://www.cqvip.com

关系规定所有的复制副本以相同的顺序执行更新操作。例如,Golding[46]系统应用 Ack Vector 来估计副本的状态。副本互换各自的 Ack Vector,并采用 Timestamp Vector 的方式更新该向量,保证副本间的数据一致性。又如,Bayou 系统[44]指定一个专门的 Pimary 负责决定所有副本更新操作的提交顺序。它为每一提交的更新请求分配一个序列号,并将请求和该序列号的对应关系传递给系统中的所有副本。副本根据该序列号顺序地提交更新请求,保证系统一致性。再如,在 Active Replication 中,所有的副本以相同的顺序接收和处理客户请求,以维持服务副本间的一致性。请求到达的一致性依赖底层的组通信系统来保证。另外,在 Deno 系统[47]中,采用一种乐观的 Quorum 一致性协议,以保证副本一致。与全序关系实现不同,偏序关系通常使用更新操作间的语义或因果关系予以实现。

当系统中某一对象副本执行更新操作时,若有其它更新操作在副本间传播,则系统产生更新冲突。更新冲突的解决很大程度依赖于应用,有时甚至要求手工解决[44]。一种通用的方法是应用时戳技术。为了检测更新冲突,系统为每一更新操作附加上更新发起者的时戳。若某一更新操作比另一操作具有更新的时戳,则无冲突,否则更新冲突出现。在 Microsoft 的 Active Directory 中,规定拥有更多更新操作的站点将赢得冲突。一旦两个站点具有相同数目的更新操作,则系统将根据它们各自的时戳来解决冲突。

针对分区网络中复制一致性问题,当前的研究主要分为两类方法:乐观法和悲观法。在乐观法中,一旦网络产生分区,各分区可以独立地执行更新请求。当分区合并时,各分区通过相互比较各自对象副本的状态,以执行一致性检测。例如,LOCUS 系统[48]采用一种基于版本号的向量方法,系统中的每个文件都拥有一个 n 维的向量,n 是包含该文件副本的节点数。向量的每一维对应相应节点上文件副本的版本号(即更新次数)。正常情况下,系统保持全连接,因此每次更新对所有的文件副本执行操作且所有的版本号相同。一旦网络产生分区,因为不同的分区允许独立地执行更新操作,从而可能导致不同分区中节点的版本号向量不同。当分区合并时,通过比较不同分区中节点拥有的版本号向量,可以检测和解决更新冲突,以实现文件副本间的一致。悲观方法通常采用基于法定多数的选举机制,以实施更新操作[47]。当某一节点实施更新前,它总是发起选举,以收集执行更新要求的足够选票。一旦获取多数选票,则赢得本次选举并实施更新,以保证对象副本间的一致性。

4.2 值故障

传统的复制协议[28-32]大都假设整个环境只存在 Fail-Stop 类型故障,极少考虑如何屏蔽值故障。从技术角度,一种屏蔽和检测值故障的通用方法是采用基于多数表决主动复制技术[10,11]。它通过收集和比较来自同一复制组内其它成员的消息,选举出与多数成员一致的结果并回送,从而达到屏蔽值故障的目的。

多数表决主动复制技术的核心是多数表决算法。具体而言,需要解决的问题包括表决过程中组成员数和多数值大小的改变。

多数表决主动复制技术的一个典型例子是 AQuA[10]。AQuA 采用主动复制通信方案,维持不同副本间的数据一致性;提供适应性多数表决算法,保证在组成员数和多数值大小动态改变时仍能正确执行表决。

除 AQuA 系统外,其它相关的研究大都集中在如何解决组成员数的改变问题。例如,Delta-4 系统[25]采用主动复制并考虑了表决过程中因为崩溃故障而导致的复制组中对象副本数的改变问题,但它忽略了表决过程中多数值大小的变化。又如,Eternal[11]系统将更加安全的令牌环组通信协议 SecureRing[49]和多数表决主动复制技术相结合,并引入基组概念,对请求/应答消息执行多数表决,以屏蔽故障。Eternal 解决了表决过程中组成员数的变化问题,但不允许应用动态改变影响表决结果的多数值大小。

4.3 故障管理

故障管理包括故障检测和恢复两方面。由于系统只有检测到故障才能执行相应的定位和恢复,因此故障检测是故障恢复的前提。

针对故障检测问题,目前主要集中在对故障监控模式的研究。按照故障信息在系统中流动的策略,有两种主要的监控模式:Push 模式和 Pull 模式[13]。在前一模式中,控制流的方向和信息流的方向匹配,被监控对象处于主动方。它周期性地发送"心跳"消息给故障检测服务,表明其存活性。一旦故障检测服务在规定的超时到达前没有收到被监控对象送回的心跳消息,则它认为该对象产生故障。而在 Pull 模式中,信息流动的方向与控制流相反,故障检测服务位于主动方。它周期性地发送消息给被监控对象,询问它是否存活。一旦被监控对象回送应答,则意味着它工作正常,否则故障检测服务怀疑它已经发生故障。

针对故障恢复问题,现有研究大都采用基于消息日志的回滚/恢复技术。它使用分段确定性模型(PDM, Piecewise Deterministic Model),并通过消息日志和确定性重做机制,实现故障进程的恢复,从而避免多米诺效应。基于消息日志的回滚/恢复技术大体分为以下两类:悲观日志法和乐观日志法。在悲观日志方法中,进程在接收到每一消息的同时将消息同步地保存到稳定的存储介质中。一旦进程产生失效,它能够重做记录在稳定存储介质中的消息,从而实现本地恢复,并无需和系统中其它进程通信。乐观日志法则采用异步方法,先将消息保存到内存中,然后再周期性地记录到稳定的存储介质中,应用进程无需阻塞等待消息日志的完成。与悲观日志法相比,乐观日志法具有较好的性能,但也产生一定的问题。如果进程在出现故障前发生消息丢失,它通常需要比较复杂的回滚/恢复机制。进程间必须通过彼此协调来确定一个乐观一致的校验点集合,并从该点实现恢复。

5 典型的可信中间件系统

5.1 Electra

Electra[15]系统是瑞士苏黎士大学的研究项目,其目标是通过扩展 ORB 核心,以支持可信性,并弥补 CORBA 规范无法提供可靠分布通信的内在缺陷,从而为分布对象应用建立一个可信的服务框架。Electra 是第一个采用动态复制技术支持可信计算服务的 CORBA 系统。

Electra 系统的最底层是 Horus[50]组通信包,负责为分布对象应用提供可靠、全序的组通信服务,以维护复制服务副本的一致性。另外,Electra 系统也对 ORB 的基本对象适配器 BOA 进行了修改和扩充,提供对象组的支持,以实现复制对象的创建、删除和状态同步。Electra 系统的主要问题是要求对 ORB 核心进行修改,难以与已有的 CORBA 标准兼容。但由于组通信和复制机制被集成到 ORB 中,因此可信性保障机制对分布应用具有较好的透明性。

5.2 Orbix+Isis

Orbix+Isis 系统[16]是第一个支持可信计算服务的商用中间件系统。它采用与 Electra 类似的策略,即通过修改 Or-

维普资讯 http://www.cqvip.com

bix的核心以实现与组通信系统Isis的有效集成。在Orbix+Isis系统中,一个服务实现类必须具体地继承一个基类,以获得可信对象组的行为。基类包括以下三种类型:Active Replication,Hot Passive Replication和Event Stream。其中,Active Replication和Hot Passive Replication允许应用对象实现不同的复制策略,而Event Stream则为应用提供基于发布/订阅模式的消息服务。另外,Orbix+Isis也采用IsR(Isis Repository)以存储服务角色的描述信息,从而允许客户应用根据服务对象的角色配置通信层。Orbix+Isis系统具有较高的效率和较好的透明性,但可移植性和互操作性较差。

5.3 AQuA

AQuA[10]系统是美国伊利诺斯大学和朗讯贝尔实验室联合研制的中间件系统,其目标是为分布应用提供一个适应性可信计算基本框架。它包括以下部分:Proteus,QuO及Ensemble/Maestro[12]协议栈等。其中,Proteus是整个系统的核心,它允许应用使用QuO[33]指定其期望的可信性需求并通过适应性重配,达到动态容错的目标。位于底层的Ensemble/Maestro协议栈负责为应用提供强大的组通信能力,以支持可靠的消息多播服务和全序的消息交付服务,从而维持复制组成员副本间的一致性。

Proteus包括可信性管理器、对象工厂及网关。其中,可信性管理器根据故障信息及应用的可信性需求确定系统的配置。驻留在每一主机的对象工厂负责创建和销毁对象副本,并为可信性管理器提供当前主机的负载及其它信息。对象网关则通过提供适当的接口实现Ensemble协议栈支持的进程级通信与IIOP消息间的有效转换。

5.4 Eternal

Eternal[11]系统是加州大学巴巴拉分校开发的可信中间件系统,其目标是为分布对象应用提供透明的可信计算基础设施。Eternal采用拦截器法,截获ORB到传输层的IIOP消息,并将其重新映射到底层可靠的组通信系统Totem[51]中,发往服务器对象。这种操作系统级的拦截调用允许Eternal以更小的开销提供支持可信计算的服务功能。Eternal系统支持主动复制和被动复制,克服了多线程CORBA内在的不确定性,引入了容错域的概念,允许外部客户通过网关访问域内的服务对象。Eternal系统主要由三部分构成:复制管理器、资源管理器和演化器。其中,复制管理器根据用户定义的可信性要求,为应用创建复制对象副本,并将这些副本部署在不同的主机中。资源管理器负责监控系统资源的使用情况,以维护初始复制副本数和最小复制副本数。演化器采用对象复制技术以实现应用对象的动态升级。另外,Eternal也使用日志恢复机制达到失效恢复的目的。

Eternal系统具有较好的可信计算服务透明性,并不要求对ORB核心进行修改和扩充,并能以非常小的开销为分布对象应用提供可信性支持。另外Eternal的Realize[35]子系统提供了多级反馈机制,支持资源的动态调度和迁移,具有环境的自适应能力。

5.5 OGS

OGS系统[17]是瑞士联邦工学院洛桑分校研制开发的可信对象中间件系统,其目标是为分布对象应用集成组通信服务,满足分布对象应用的可信性需求。它通过在ORB之上为应用定义相应的IDL接口,实现了一组通用的服务,如消息传递、多播通信、失效监控及一致性维护等,以支持对象容错。其中,消息服务实现了将消息映射到传输层的相应机制;多播通信服务为应用提供可靠的多播通信原语;一致性服务为多播通信施加了相应的全序限制;故障监控服务负责为崩溃对象提供检测机制。OGS系统的主要问题是为了支持对象组操作,复制对象必须从一个公有的IDL接口派生,导致可信计算服务透明性的丧失并减少了服务实现方的灵活性。

5.6 ROAFTS

ROAFTS[18]是加州大学埃尔文分校独立开发的自适应可信中间件系统。它采用Orbix为应用提供分布对象平台的支持,其目标是通过透明地监控和自适应重配系统资源,为复杂的实时关键业务系统提供可信性支持。ROAFTS的底层是TMO(Time-trigger Message-trigger Object)对象,为应用提供实时性支持。TMO对象是对传统对象模型的扩展,它通过为对象提供时间触发的方法,以保证应用在时限到来之前完成相应的任务。ROAFTS主要包括:自适应管理器、TMO管理器、容错策略服务器、资源分配体、网络重配器和系统监控器。每一部分都完成特定的功能如自适应决策、TMO对象管理、容错策略的选择、系统资源重配和错误检测等。它们彼此协作,以适应环境和用户要求的变化,实现资源管理和自适应容错。为了支持自适应决策,ROAFTS提供了3个数据库:环境数据库、内部状态数据库和用户需求数据库,分别记录了环境状态信息、系统最近的失效历史和系统资源状态信息。自适应管理器使用这些数据库进行自适应决策并将相关信息转发给容错策略服务器和资源分配体。容错策略服务器支持DRB(Distributed Recovery Block),RB(Recovery Block)和例外处理,以满足不同的软硬件可信性需求。

ROAFTS具有清晰的层次结构和较好的环境自适应能力,但它主要针对实时关键业务系统,并要求对TMO对象提供支持。

5.7 DOORS

DOORS[13]是Lucent贝尔实验室的研究项目,它将支持可信计算服务的机制和策略作为ORB之上的通用服务对象加以实现,从而为分布对象应用提供可信性支持。DOORS主要侧重于被动复制策略,并未对组通信和虚同步模型提供支持。它允许应用开发者选择复制策略、可靠性级别、故障检测机制和恢复策略。复制管理器是DOORS系统的核心,负责决定复制对象的初始位置、对象激活并在对象失效时执行迁移策略。另外,复制管理器也维护了一个状态池,记录系统中的复制对象数、主机位置、对象状态和特定主机出现故障的次数。故障检测服务采用层次结构,分为两级:Super FaultDetector和FaultDetector。FaultDetecotor执行对象级故障检测,Super FaultDetector则负责检测主机级的故障。日志恢复机制定期地执行检查点操作,保证失效对象的恢复。

DOORS系统具有较好的兼容性,不要求对ORB核心进行更改,完全遵循CORBA标准,但可信性保障机制缺乏对应用的透明性。

结束语 本文对可信中间件技术的研究进行了全面的分析和总结。首先介绍了涉及可信中间件的一些基本概念,然后从研究层次、实现方式及研究所针对的问题等方面系统地分析了现有的研究工作。随后对可信中间件系统关键问题的研究现状进行了详细讨论。另外,还介绍了一些典型的可信中间件系统。

我们认为,可信中间件系统的研究将在以下几方面继续深入:

研究适用于分区网络环境的复制一致性协议,保证复制更新的全局一致性;针对系统中可能同时出现的崩溃故障和值故障,研究可屏蔽崩溃故障和值故障的复制技术;针对现有故障检测服务在灵活性、伸缩性和扩展性方面的不足,研究适用于不同粒度实体的故障检测方法;为可信中间件系统中集

维普资讯 http://www.cqvip.com

成安全保障机制，构建安全可信的分布对象应用系统；探索可以同时满足实时性和可信性需求的中间件技术，保证系统在遭遇无法预料的故障或不可预测的故障恢复时间时仍能提供正确的服务。

分布计算应用领域的不断拓展，推动了分布应用可信性需求的日益增长，对中间件系统提出了新的挑战。如何尽量保持对应用的透明性，减少应用开发者对可信性的关注，继承大量遗留的应用，将是分布应用对可信中间件系统长期而主流的需求。

参 考 文 献

[1] OMG. The Common Object Request Broker: Architecture and Specification V3.0.2, Object Management Group, Feb. 2002

[2] Horstmann M, Kirtland M. DCOM Architecture. msdn. http://microsoft.com/library

[3] Pitt E, McNiff K. Java. rmi: The Remote Method Invocation Guide. Addison Wesley, 2001

[4] Laprie J C. Dependability: Basic Concepts and Terminology, Vienna Springer-Verlag, 1992

[5] Karamanolis C T, Magee J N. Configurable Highly Available Distributed Services// Proceedings of 14th Symposium on Reliable Distributed Systems. Sep. 1995

[6] Powell D. Failure Mode Assumptions and Assumption Coverage // Proceedings of the 22nd Annual International Symposium on Fault-Tolerant Computing (FTCS '92). 1992:386-395

[7] Kim K H. ROAFTS: A Middleware Architecture for Real-Time Object-Oriented Adaptive Fault Tolerance Support// Proceedings of HASE '98 (IEEE CS 1998 High-Assurance Systems Engineering Symp.). Washington D C, Nov. 1998:50-57

[8] Kopetz H, et al. Distributed Fault - Tolerant Real-Time System: The MARS Approach. IEEE Micro, 1989,9(1):25-40

[9] Stankovic J, et al. The Spring System: Integrated Support for Complex Real-Time Systems, 1998

[10] Ren Y J, Courtney T, Cukier M, et al. AQuA: An Adaptive Architecture that Provides Dependable Distributed Objects. IEEE Transactions on Computers, 2003, 52(1):31-50

[11] Narasimhan P. Transparent Fault Tolerance for CORBA. PhD thesis. University of California at Santa-Barbara, 1999

[12] Vaysburd A, Birman K P. The Maestro Approach to Building Interoperable Distributed Applications with Mutiple Execution Styles. Theory and Practice of Object Systems, 1998, 4(2):73-80

[13] Natarajan B, Gokhale A, Yajnik S, et al. DOORS: Towards High-performance Fault Tolerance CORBA// Proceedings of the 2nd Distributed Applications and Objects (DOA) Conference. Antwerp, Belgium, Sep. 2000:21-23

[14] Kalbarczyk Z, Iyer R K, Bagchi S. Chameleon: a Software infrastructure for Adaptive Fault Tolerance. IEEE Transactions on Parallel and Distributed Systems, 1999, 10(6):560-579

[15] Maffeis S, Schmidt D C. Constructing Reliable Distributed Systems with CORBA. IEEE Communications Magazine, 1997, 35(2):56-60

[16] Iona Technologies PLC. Orbix Programmer's Guide. Oct. 1997

[17] Guerraoui R, Eugster P, Felber P, et al. Experiences with Object Group Systems// Software: Practice & Experience. 2000, 30(12):1375-1404

[18] Kim K H, Ishida M, Liu J. An Efficient Middleware Architecture Supporting Time-Triggered Message-Triggered Objects and an NT-based Implementation// Proc. ISORC '99 (IEEE CS 2nd Int'l Symp. on Object-oriented Real-time distributed Computing), May 1999:54-63

[19] Fabre J C, Perennou T. A Metaobject Architecture for Fault-Tolerant Distributed Systems: The FRIENDS Approach. IEEE Transactions on Computers, 1998, 47(1):78-95

[20] Narasimhan P, Dumitras T A, Paulos A M, et al. MEAD: Support for Real-Time Fault-Tolerant CORBA, Concurrency and Computation: Practice and Experience, 2005, 17(12):1527-1545

[21] Maffeis S. Piranha - A CORBA Tool for High Availability. IEEE Computer, 1997, 39(4):59-66

[22] Veritas. Veritas FirstWatch. www.veritas.com/us/products/firstwatch, 2000

[23] Huang Y, Kintala C. Software Implemented Fault Tolerance: Technologies and Experience// 23rd International Symposium on Fault-tolerance Computing (FTCS). Toulouse, France, Jun. 1993:2-10

[24] MSCS. Microsoft NT Server Edition. www.microsoft.com, 1998

[25] Powell D. Distributed Fault Tolerance— Lessons Learnt from Delta-4, IEEE Micro, 1994, 14(1):36-47

[26] Powell D, Arlat J, Beus - Dulic L, et al. GUARDS: A Generic Upgradable Architecture for Real-Time Dependable Systems. IEEE Transaction on Parallel and Distributed Systems, 1999, 10(6):580-597

[27] Nguyen - Tuong A. Integrating Fault - Tolerance Techniques in Grid Applications. PhD thesis. University of Virginia, 2000

[28] Schneider F B. Replication Management Using the State-Machine Approach. Mullender S. ed. Distributed Systems, ACM Press, 1993:169-197

[29] Budhiraja N, Marzullo K, Schneider F B, et al. The Primary-Backup Approach// Mullender S, ed. Distributed Systems. Second Edition. ACM Press Books, Chapter 8, 1993, 199-216

[30] Bernstein P A, Hadzilacos V, Goodman N. Concurrency Control and Recovery in Database Systems. Reading, MA: Addison Wesley, 1987

[31] Thomos R H. A Majority Consensus Approach to Concurrency Control for Multiple Copy Databases, ACM Transactions on Database Systems, 1979, 4(9):180-209

[32] El Abbadi D A A. The Tree Quorum Protocol: An Efficient Approach for Managing Replicated Data// Proceedings of the 16th VLDB Conference. Brisbane, Australia, 1990

[33] Krishnamurthy Y, Kachroo V, Da K, et al. Integration of QoS-Enabled Distributed Object Computing Middleware for Developing Next-Generation Distributed Applications// Proceedings of the ACM SIGPLAN Workshop on Optimization of Middleware and Distributed Systems (OM 2001). Snowbird, Utah, Jun. 2001

[34] Krishna A, Schmidt D C, Raman K, et al. Optimizing the ORB Core to Enhance Real-time CORBA Predictability and Performance// Proceedings of the Submitted to the 5th International Symposium on Distributed Objects and Applications (DOA). Catania, Sicily, November 2003

[35] Melliar-Smith P M, Moser L E, Kalogeraki V, et al. Realize: Resource Management for Soft Real-time Distributed Systems// Proceedings of the DARPA Information Survivability Conference. Hilton Head, SC, Jan. 2000:281-293

[36] Li B, Xu D, Nahrstedt K. An Integrated Runtime QoS-aware Middleware Framework for Distributed Multimedia Applications. ACM Multimedia Systems Journal, 2002, 8(5): 420-430

[37] Mohapatra S, Venkatasubramanian N. A Distributed Adaptive Scheduler for QoS Support in Compose|Q// Proc of the IEEE Workshop on Object-Oriented Real-Time Dependable Systems. Jan 2002:326-333

[38] Guessoum Z, Ziane M, Faci N. Monitoring and Organizational-Level Adaptation of Multi-Agent Systems // AAMAS' 04. ACM, New York City, July 2004:514-522

[39] Kumar S, Cohen P R, Levesque H J. The Adaptive Agent Architecture: Achieving Fault-Tolerance Using Persistent Broker Teams// Proceedings of the 4th International Conference on Multi-Agent Systems (ICMAS2000). Boston, MA, July 2000

[40] Morgan G, Shrivastava S, Ezhilchelvan P, et al. Design and Implementation of a CORBA Fault-Tolerant Object Group Service // Proceedings of the Second IFIP WG 6.1 International Working Conference on Distributed Applications and Interoperable Systems. Helsinki, Finland, Jun. 1999

[41] Marchetti C, Mecella M, Virgillito A, et al. An Interoperable Replication Logic for CORBA Systems// Proceedings of the International Symposium on Distributed Objects and Applications. Sep. 2000:7-16

[42] Friedman R, Hadad E. FTS: A High Performance CORBA Fault Tolerance Service// Proceedings of IEEE Workshop on Object-oriented Real-time Dependable Systems. San Diego, CA, Jan. 2002

[43] Pu C, Leff A. Replica Control in Distributed Systems: An Asynchronous Approach// Proc. of the ACM SIGMOD Intl Conference on Management of Data. May 1991:377-386

[44] Terry D, Theimer M, Petersen K, et al. Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System// 15th Symp. on Operating Systems Principles (SOSP). Copper Mountain, CO, Decem, 1995:172-183

[45] Rubel P. Passive Replication in the AQuA System. Master's thesis. University of Illinois at Urbana-Champaign, 2000

[46] Golding R. Weak-Consistency Group Communication and Membership. PhD thesis. University of California at Santa Cruz, Dec. 1992

[47] Keleher P. Decentralized Replicated-Object Protocols// Proc. of the 18th ACM Symp. on Principles of Distributing Computing. Apr. 1999

[48] Parker D, Popek G, Rudisin G, et al. Detection of Mutual Inconsistency on Distributed Systems. IEEE Trans. on Software Engineering, 1983, SE-9(3):240-247

[49] Kihlstrom K P, Moser L E, Melliar-Smith P M. The Secure-Ring Protocol s for Secure Group Communication// Proceedings of the IEEE 31st Hawaii International Conference on System Sciences. Vol 3. Kona, Hawaii, Jan. 1998:317-326

[50] Renesse R V, Birman K P, Maffeis S. Horus: a Flexible Group Communication System. Communications of the ACM, 1996, 39(4):76-83

[51] Agarwal D A, Melliar-Smith P M, Moser L E, et al. The Totem multiple-ring ordering and topology maintenance protocol. Transactions on Computer Systems, 1998, 16(2)