一种基于自适应复制的多 Agent 容错模型

顾佳伟
(同济大学 电子与信息工程学院,上海 200092)

摘 要:为了构造和部署大规模的多 agent 系统,人们必须找到并解决其基本问题,其中之一就是可能存在的局部性系统故障。这也就意味着,容错对于大规模多 agent 系统来说,是一个无法回避的主题。文中讨论了这类问题并且提出了一种多 agent 系统的容错方法。最先的想法是将复制策略运用到 agent 中,对处于危急状态的 agent 进行复制从而避免系统故障,但是由于 agent 的危急性会在执行过程中演变,并且 agent 的可用资源是绑定的,所以需要动态以及自动地调整 agent 的复制体个数,从而最大化它们的作用和可靠性。文中将描述评估某个 agent 危险性的方法以及相关机制,并且决定使用何种策略(如:主动复制,被动复制)以及如何将其参数化(如:复制的个数)。

关键词:容错;多 agent 系统;复制;自适应;相互依赖;危险性
中图分类号:TP302.8 **文献标识码**:A **文章编号**:1673-629X(2007)08-0140-04

An Adaptive Multi-Agent Fault-Tolerant Architecture

GU Jia-wei
(College of Electronics and Information Engineering, Tongji Univ., Shanghai 200092, China)

Abstract: To construct and deploy large-scale multi-agent system, must address and solve some fundamental issues, one of which is the possibility of partial failures. This means that fault tolerance is an inevitable topic for the large-scale multi-agent system. So, in this paper, a new approach of fault tolerance in multi-agent systems is discussed. First introduce the notion of replication to the agent system and replicate the critical agents to protect the system from the failure. However, as the agent criticality will evolve in the course of executing and the available resources are bounded to the certain agents, need to dynamically adapt the number of replica agent in order to maximize their reliability and availability. Meanwhile, this paper will include the approach and mechanism to evaluate the agent criticality, how to decide the replica strategy (active or passive) and how to parameter them (numbers of replicas).

Key words: fault-tolerance; multi-agent system; replica; adaptive; interdependence; criticality

0 引 言

存在局部错误性是分布式应用的一个基本特征。容错研究团体(fault-tolerance research community)也开发出一些解决方案(包括算法和体系结构),这些解决方案大多数是基于复制的理念,用于诸如数据库等应用中。但是,这些方案中的复制大多是明确和静态的复制,在设计阶段就已经确定。在这样的方法中,设计者必须一开始就明确指定哪些服务器需要很高的健壮性,也要决定运用何种策略(主动或是被动)以及如何配置复制个数和位置等问题。

而新的合作系统,比如交通控制系统、协同工作或是电子商务系统比一般的系统更具规模性和动态性,因此,想要提前指出系统中最具危险性的软件组件是很困难甚至是不切实际的。另外,组件的危险性会随着系统的运行而发生变化,必须最优化地使用有限的复制资源。这类合作系统现在越来越多地被设计成一组自主的交互的实体,即 agent,通过 agent 的交互与合作,构成一个多 agent 系统(multi-agent system)[1]。在这样的应用中,agent 的角色和相对重要性可能会在计算、交互与合作的过程中发生很大变化,所以 agent 必须具备变换角色和策略的能力,而新的 agent 也可以加入或者离开该应用。此外,此类应用也可能是大规模的,这就决定了这类应用更易产生不稳定性,也就更需要特定的机制使其自适应地提高自身的可靠性。

文中的方法使得多 agent 系统具有了动态识别最危急状态的 agent 并且决定使用何种策略提高其可靠性的能力,有点类似于"负载平衡",但仅仅用于提高 agent 的可靠性。我们只是想要在最需要的时候可以自动并且动态地提高 agent 的可靠性(大多数情况下是通过复制机制)。而为了动态地提高 agent 的可靠性,

收稿日期:2006-11-26
作者简介:顾佳伟(1983-),男,江苏太仓人,硕士研究生,研究方向为智能分布式系统。

需要用到各种级别的信息,比如系统级的通讯负载,以及应用级的全局任务或者计划。

1 相关研究

目前对于多 agent 系统的容错技术有很多,这些方法又可以大体分为两类。第一类着重于考虑多 agent 系统中某一 agent 的可靠性,这类方法主要通过追踪并捕获 agent 之间的通信、交互与合作中的严重错误,并进行处理。而第二类方法则关注于移动 agent 的可靠性,相对于多 agnet 系统来说,移动 agent 更容易出现安全问题,也更难保证其可靠性,此类问题已经超越了文中所要论述的范围,文中主要讨论第一类方法。

S. Hagg 引进的哨兵机制主要用于避免 agent 进入一些不被期望的状态[2]。哨兵即代表了这种多 agent 系统的控制结构的核心,他们需要构建每一个 agent 的模型,并对其进行监控然后对错误作出及时的反映。设计者将每个哨兵关联到多 agent 系统中的每一个功能上,并监控实现该功能的那些 agent,然后,通过分析自身对于这些 agent 的信念,哨兵就可以在一个错误发生的时候捕获它。虽然表面上给多 agent 系统加入哨兵是个很好的容错方法,但是,哨兵中也存在多 agent 系统的缺陷,而且,作为解决问题的 agent,哨兵本身也被包涵在了整个 agent 系统运行和容错过程中间。

A. Fedoruk 和 R. Deters[3]提出利用代理来使得 agent 的复制过程透明化,也就是说,让一个 agent 的复制体对于其他 agent 来说,像其本身实体一样进行行为活动,复制体的所有状态由代理来管理,群组间所有外部和内部的通信都重定向到代理上。但是,这样就会导致代理的工作负荷增加,因为它要扮演一个类似中枢的角色。为了使代理更为可靠,他们提出了为每一个群组 agent 建立一个代理层,同时也提到了其中存在的读/写一致性问题、资源加锁等等[4]。然而,这个方法却缺乏了灵活性和重用性,尤其是对于复制控制的灵活性和重用性。而基于 FIPA-OS 工具的相关实验也没有提供任何的复制机制,相应的复制都是由设计者在运行之前实现好的。

Kaminka et al[5]改进了一种监视方法,从而使其可以发现并恢复错误。他们利用了 agent 精神状态之间的关系模型,采用了一种基于阶段性计划识别的方法(procedural plan-recognition based approach)来发现不一致性。但是,这种改进仅仅是结构上的,关系模型是会实时改变的,计划的上下文条件却是静态的。他们的主要假设就是任何错误都来自于信念上的不完整。由于该监视方法与 agent 的认知有关,因此设计这样的多 agent 系统是相当复杂的,而且该方法也并不适用于开放式的和自适应的多 agent 系统。

许多工具包(如文献[6]和[2])自带一些用于构造可靠应用的复制工具。但是,它们中大多数并不太适用于实现大规模、自适应的复制机制。例如,有些工具可以在计算过程中更改某些策略,但是并没有相应地执行该策略的指导,而且,这样的策略改变也是设计者在程序运行前制定好的。另外,由于各个功能的组织结构是让用户去指定的,这也大大增加了设计大规模软件应用的复杂度。

下文将介绍一种新的 agent 架构,其中包含了自动以及自适应的复制控制。

2 监控多 agent 架构

监控就是获取必要的信息从而动态且自动地进行必要的复制。这些必要的信息可能是诸如通讯负载或者处理时间这样的标准度量,也可能是 agent 角色或者 agent 的相互依赖这样的 agent 固有特征。

目前大多数现有多 agent 体系结构中,观察机制都是集中化的。而获得的数据通常只是在 agent 离线后用于解释和提升系统的行为能力的,因此,这种集中化的观察结构并不适合大规模的复杂系统,因为这类系统要求能够实时分析观察到的数据以使得该多 agent 能够适应它所处环境的演化。

所以,在这里提出一种将观察机制分布化的方法以提高其性能和健壮性。该分布式机制依靠一种反映式的 agent 组织,这种 agent 有两个任务:一是观察领域 agent(domain agent)并控制它们的复制;二是构建全局信息以减少通信。这两种任务分别指派给两种 agent 执行:(领域)agent 监视器和主机监视器。每个 agent 监视器和领域 agent 关联,而主机监视器则与每个主机相关联(见图 1)。

这些监视型 agent(agent 监视器和主机监视器)是分层次组织的。每个 agent 监视器仅仅和一个主机监视器通信。而主机监视器之间可以交换本地信息从而构建系统全局信息(全局消息数、全局信息交换量等等)。

每个 agent 监视器一经创建就要向主机监视器注册,然后才可以接受与其相关联的领域 agent 的有关数据和事件。在一段△时间间距后,主机监视器会发送收集到的事件和数据到相对应的 agent 监视器。然后这些 agent 监视器调用自适应算法(见 3.2)。当一个节点的弧的数量发生很大变化的时候(即相互依赖性发生很大变化,见后文),agent 监视器便会告知主机

监视器,然后由后者通知其他主机监视器,从而更新全局信息。之后,其他 agent 监视器也会依次从各自主机监视器处获知全局信息的重大变化。

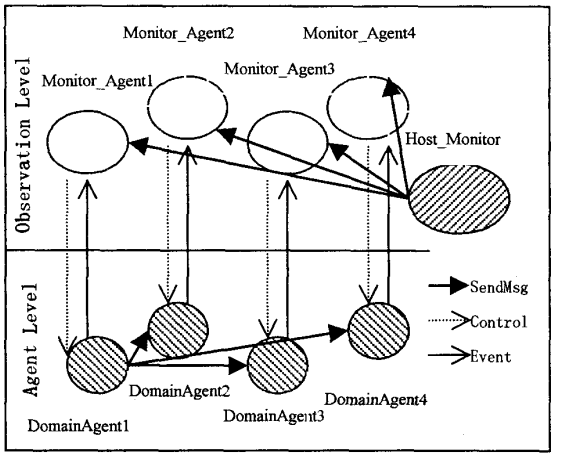

图 1 多 agent 架构

3 agent 危险性

在多 agent 系统中,每一个 agent 都是一个自主的实体。但是单个 agent 往往不具有需要的资源和能力而必须转求其他 agent 来提供。于是,人们就引入了相互依赖图(Interdependence graphs)[7~10]来描述 agent 之间的相互依赖性。这些依赖图都是由设计者在多 agent 系统执行之前设计好的。但是,复杂的多 agent 系统常常伴有突现的结构(emergent structures[8])的产生,因此,并不能在运行之前就静态地设计好这类多 agent 系统。

因此,在文中的架构中,多 agent 系统是由一张可以反映突现组织结构的相互依赖图来表示的。而且也可以通过解释这种结构来定义每个 agent 的危险性。

3.1 相互依赖图

将每个领域 agent 关联到一个节点。这些节点的集合即构成了相关依赖图(见图 2),用有向图(N,L,W)来表示它。N 表示图中的节点集合,L 表示弧的连接,而 W 表示权值的集合。

$$N = \{N_i\}_{i=1,n} \tag{1}$$

$$L = \{L_{i,j}\}_{i=1,n,j=1,n} \tag{2}$$

$$W = \{W_{i,j}\}_{i=1,n,j=1,n} \tag{3}$$

$L_{i,j}$ 表示节点 N_i 和 N_j 之间的连接,而 $W_{i,j}$ 则是连接 $L_{i,j}$ 上的实际数值,它反映了该连接关联的两个 agent($agent_i$ 和 $agent_j$)之间相互依赖的重要性。因此,可以充分利用这些权值来检测哪条连接上的负载过重或者判断系统是否过度依赖于较少的一些 agent。

这样一来,一个节点就与一个可能包含系统中所有节点的节点集相联系了起来,并且,这个节点集不是静态的,当新的领域 agent 加入或者原有的领域 agent 消失时,这个节点集就会被改变。

下文将介绍相互依赖图的自适应算法。

3.2 自适应算法

agent 之间的依赖性可以用很多参数来定义,例如通讯负载、已执行的任务数、agent 角色等等。一个自适应算法列出了相互依赖图中自适应机制的概要。这种自适应依赖于本地信息(通讯负载)和全局信息(本地信息集合)。而每个 agent 监视器利用一种自适应算法来管理与之关联的节点。

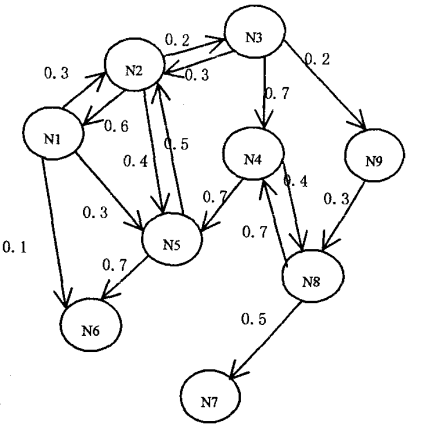

图 2 相互依赖图示例

首先,假设一个时间间隔 Δt。在每个 Δt 之后 agent 监视器就会被激活,然后这些 agent 监视器按照步骤执行自适应算法(见下文)。而领域 agent 依然连续的按照原先目标执行。

下面就介绍一种计算两个 agent 之间相互依赖的算法,其原理是分析 agent 之间交换的消息数量来计算他们的相互依赖。

该算法需要依赖全局消息发送总量。这个总量 $NbM(\Delta t)$ 是由主机监视器通过如下公式计算获得:

$$NbM(\Delta t) = \text{opl}(NbM_{1,2}(\Delta t),\cdots,NbM_{n,n-1}(\Delta t)) \tag{4}$$

其中:$NbM_{i,j}$ 是在时间段 $\triangle t$ 之内 $agent_i$ 向 $agent_j$ 发送的消息数总量。opl 是一种聚合操作,这里取均值。

自适应算法:

1:for each j different of I do

2: Calculate:

$$\text{MyNbM} = (\text{NbM}_{i,j}(\Delta t) - \text{NbM}(\Delta t))/\text{NbM}(\Delta t) \tag{5}$$

3: Update the weights by using the following rule:

$$\text{W}_{i,j}(t+\Delta t) = \text{W}_{i,j}(t) + \text{MyNbM} \tag{6}$$

4:end for

4 资源管理

对于 agent 危险性的分析使我们可以定义 agent 故障对于整个多 agent 系统行为和可靠性的影响。提出了利用 agent 之间相互依赖来定义 agent 的危险性,并用每个 $W_{j,i}$ 的和值 $W_{j=1,n,i}$ 来表示 agent$_i$ 的危险性。然后,就可以决定每个 agent 的复制数量了。该数量决定于以下各项:

wi:agent$_i$ 的危险性

W:领域 agent 的危险性综合

rm:设计者引入的最小复制数量

Rm:可用资源数,它决定了可能的最大并发复制数量

agent$_i$ 的复制数量 nb_i 可以如下确定:

$$nb_i = \text{rounded}(rm + wi * Rm/W) \tag{7}$$

这种方法原理很简单,所有资源都被无差别地对待。例如,它并不理会不同主机的的错误率。笔者建议用经济学的理念来处理不同类别资源管理的问题和 agent 复制问题。因为资源集合确实会随着新的主机的加入或者移除而动态变化。而且,由于多 agent 系统是开放式的,所以 agent 完全可以容易地被添加或者移除。

文献[7]展现了利用经济学理念处理资源分配问题的优势。所以,笔者建议利用一个经济学模型来提高危险性 agent 复制分配的效率。先定义了一些有用的参数:资源代价,预算以及 agent 监视器与主机监视器分配资源时的协商行为。另外,这些 agent 的交互也需要遵循共同约定的网络协议。每个主机都提供一定数量的资源,这个数字可以由主机拥有者设定和改变。同时,设计者也指定了每个资源的初始代价,而这些代价值也可以在运行中由主机监视器来更新。在时刻 t,主机 i 中的资源代价定义如下:

$$CM_i(t) = CM_i(t_0) * (1 - pp_i(t)) \tag{8}$$

其中:$pp_i(t)$ 是主机 i 在时刻 t 出现错误的可能性,这个可能性是由观察模型给出的。$CM_i(t_0)$ 是主机 i 的初始资源代价。

$$pp_i(t_0) = 0$$

此外,每个 agent 的危险性数值也反映了其与系统的相关性(即代价)。因此在时刻 t,agent 监视器 j 可以通过与之相关联的领域 agent 的危险性 $W_j(t)$ 计算出它的(代价)预算 $B_j(t)$:

$$B_j(t) = W_j(t) * CM(t)/W(t) \tag{9}$$

$$W(t) = \sum_{i=1,n} W_i(t) \tag{10}$$

$$CM(t) = \sum_{i=1,m} CM_i(t) * Nb_i \tag{11}$$

其中,n 是 agent 的数量,m 是主机的数量,Nb_i 是主机 i 的所有资源数。

模型中资源的分配基于约定的网络协议。agent 监视器(发起者)先发送一个请求到主机监视器,若主机监视器当前有可用资源,则回应该 agent 监视器,它会发送一个包含其资源利用代价的请求给发起者。该发起者通过某种策略决定选择一个最适合的资源。

agent 监视器利用两种标准评估各类请求:(1)两个主机之间的通讯时间(agent 监视器本身所在的主机以及参与通讯的另一个主机);(2)资源的利用代价。其中,通讯时间是非常重要的,因为它可以影响到整个系统的运行效率。例如,重要 agent 和其复制体之间的高通讯延迟会导致网络的超负荷运行。而资源利用代价则显得不那么重要。因此,有时候很难比较(1)将代价非常昂贵的复制体放在最为可靠的主机上或者是(2)将两个复制体放在两台普通的主机上这两种方案孰优孰劣。

另外,agent 监视器可能使用多种策略来选择某个请求。例如,它们会依照与相应主机的通讯时间延迟将不同的请求分为低级、中级和高级三类。然后,它们会根据自己的预算选择通讯延迟最好的资源。

5 结　论

提出了一种新的提高多 agent 系统可靠性的方法。它是基于相互依赖性的概念的,每个 agent 的危险性由它与其他 agent 之间的相互依赖性决定。然后就可以针对其危险性的不同进行复制,从而使系统在现有资源和代价下,达到最大可用性和最高的可靠性。文中只给出了部分思路和想法,还没有付诸于实现,这也是今后的工作之一,相信将该思想应用于大规模实时系统将会带来系统可靠性和健壮性的有效提升。

参考文献:

[1] Wooldridge M. 多 Agent 系统引论[M]. 北京:电子工业出版社,2003.

[2] van Renesse R, Birman K, Maffeis S. Horus: A flexible group communication system[J]. Communications of the ACM, 1996, 39(4): 76 - 83.

[3] Fedoruk A, Deters R. Improving fault - tolerance by replicating agents[C]//In AAMAS2002. Bologna, Italy: ACM Press, 2002: 373 - 744.

[4] Silva L, Batista V, Silva J. Fault - tolerant execution of mobile agents[C]//In International Conference on Dependable Systems and Networks. New York: IEEE Computer Society, 2000: 135 - 143.

[5] Kraus S, Subrahmanian V, Tacs N C. Probabilistically surviv-

(下转第 147 页)

现在国内入侵检测厂家和产品都很多,且比较成熟,可以根据具体需要进行选配。

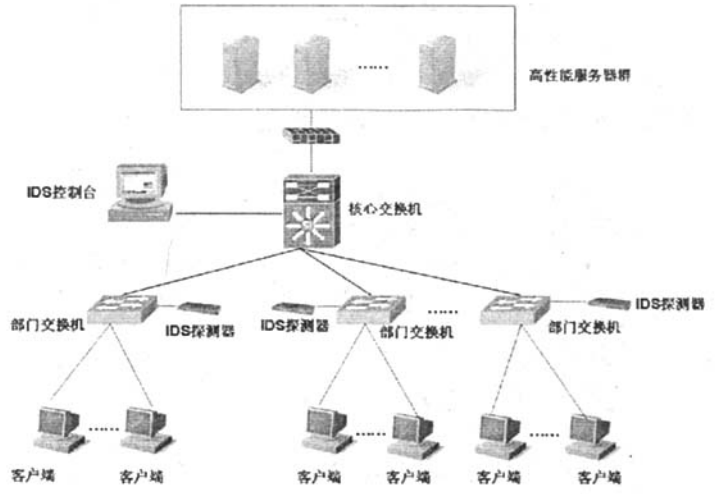

图 5　入侵检测系统方案结构图

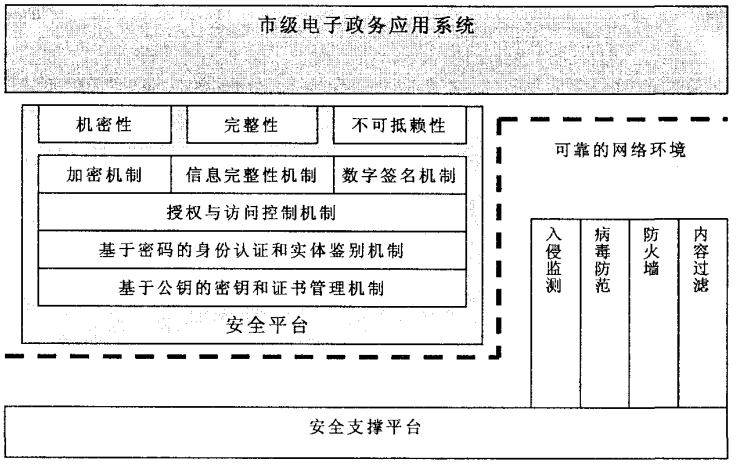

图 6　安全平台和安全支撑平台关系

3　系统总体安全性

市级电子政务系统安全平台和安全支撑平台共同为市政府电子政务建立安全、可靠的基础和环境。其中,安全平台以密码技术为核心,为电子政务应用系统提供统一的安全、可靠和可信的服务;安全支撑平台以 PDR(保护、监测、响应)为模型,为网络环境提供入侵检测、病毒防范、防火墙和信息过滤等安全功能。

市政府电子政务系统安全平台和安全支撑平台安全功能如图 6 所示。

4　结束语

市级电子政务安全平台,为市政府电子政务应用系统构建了以密码技术为基础的统一的安全平台,提供了统一的身份认证、授权与访问控制、保密性、完整性、抗抵赖等安全服务和安全机制。为政府构筑政务信息平台,实现政府网上信息交换、信息发布和信息服务提供安全保障。

参考文献:

[1] 李广乾. 电子政务及其国外发展[EB/OL]. 2006-06. www.chinainfo.gov.cn.

[2] 秦天保. 电子政务安全体系结构研究[J]. 计算机系统应用,2006(1):8-11.

[3] 濮小金. 电子政务[M]. 北京:机械工业出版社,2005:117-181.

[4] Lloyd S. Understanding the Public-Key Infrastructure: Concepts, Standards, and Deployment Considerations[M]. Carlisle: Macmillan Technical Publishing, 2004:40-42.

[5] Edir A. Economics of Information Security [J]. IEEE Security & Privacy, 2005(2): 1233-1238.

(上接第 143 页)

able MASs[C]//In Eighteenth International Joint Conference on Artificial Intelligence (IJCAI'03). Acapulco, Mexico: [s.n.], 2003:789-795.

[6] Guerraoui R, Garbinato B, Mazouni K. Lessons from designing and implementing GARF[C]//In Object-Based Parallel and Distributed Computation, number 791 in LNCS. London, UK: Springer-Verlag, 1995:238-256.

[7] Castelfranchi C. Decentralized AI, chapter Dependence relations in multi-agent systems[M]. Amsterdam, The Netherlands: Elsevier, 1992.

[8] Sichman J S, Conte R. Multi-agent dependence by dependence graphs[C]//In AAMAS2002. Bologna, Italy: ACM, 2002:483-490.

[9] Sichman J S, Conte R, Demazeau Y. Reasoning about others using dependence networks[C]//In Actes de Incontro del gruppo AI * IA di interesse speciale sul inteligenza artificiale distribuita. Roma, Italy: IP/CNR & ENEA, 1993.

[10] Sichman J S, Conte R, Demazeau Y. A social reasoning mechanism based on dependence networks[C]//In Proceedings of E-CAI'94-European Conference on Artificial Intelligence. Amsterdam, The Netherlands: John Wiley and Sons, 1994.