一种基于移动窗口的移动智能体容错机制

A Window-Based Fault-Tolerant Mechanism for Mobile Agent

马俊涛　刘积仁　刘晓明

（东北大学软件中心研究部　沈阳 110006）

Abstract Reliability is an critical factor of mobile agent system, this paper describes a window-based fault-tolerant mechanism, which can make tradeoff between performance and reliability, so is applicable to wide range of network environments. Failure recovery procedure of it is also discussed.

Keywords Mobile agent, Fault tolerant

1 引言

分布式计算是当今网络技术的主要研究领域之一，其发展已经历了以下两个阶段：(1)以 OSF/DCE 为代表的过程式分布计算环境，它采用远程过程调用(RPC)的通信机制，提供了基于过程的透明访问；(2)以 OMG/CORBA[4]为代表的分布式对象技术，基于对象请求代理(ORB)实现了异构环境中分布对象间的透明访问。

以上阶段为分布式计算的成熟和标准化做出了突出贡献，但仍存在需要解决的问题，比如：只提供被动服务，CORBA 远程资源的互操作通过"对象引用"实现，异步通信能力有限。这些不足难以满足某些特殊应用领域(如移动计算)的需要。

移动智能体(mobile agent)本质上是第三阶段的分布式计算技术，基于此视角移动智能体可定义为在网络上具有主动性和移动性的分布式对象，它具有如下的技术特征：(1)"对象传递"通信机制；(2)完善的异步计算机制；(3)资源的优化；(4)更强的并行性和非确定性。OMG 正试图制订移动智能体服务规范[3]并将其纳入 CORBA 标准的后续版本。

移动智能体引入了新的思想和方法，同时也提出了一些亟待解决的问题，如安全性、容错性和生命周期控制等，本文重点分析移动智能体容错需求并对此提出了一种基于窗口的容错机制，详细讨论其工作原理和失效恢复过程，并做出简单的性能评价。

2 移动智能体系统容错分析

传统的应用程序相对静态地运行在宿主环境中，容错性并不是突出的问题，但是在移动智能体(以下简称 MA)系统中却存在迫切的容错性需求。MA 在节点间的转移和执行是典型的串行过程，而作为过程载体的网络环境具有很强的复杂性和非确定性，其中的任一环节的意外都会造成整个过程链的中断，如网络故障、主机故障、长时间关机等。MA 任务越复杂、经过的站点越多，那么出现各种意外的概率就越大，在广域网尤其是 Internet 环境中此概率是不允许忽略的。

MA 任务求解过程中有以下可能造成崩溃的三个环节：

(1)传输过程：传输过程可能由不同介质承载，如局域网线缆、公用电话网、微波、红外线等，这些传输介质的速度、误码率和稳定性存在相当大的差异。传输过程中的误码或线路中断会导致 MA 的崩溃。

(2)服务设施：MA 经过的节点可能是不同的计算机系统，某些计算机系统有良好的容错性能，但某些系统(如微机)则容错性能很差。MA 任务的串行性使整个过程链的容错性比其中最差的节点还要差，属于典型的"灾难共享"模式。服务设施的下列行为都能够造成 MA 的崩溃或失效：系统崩溃或掉电；主机的恶意破坏；长时间停机，因时限而失效。

(3)MA 自身代码：MA 设计和实现的缺陷导致

马俊涛　博士生，研究方向智能体，组合软件工程。刘积仁　博士导师，研究兴趣为分布式多媒体，软件重用等。

特定情境下突然崩溃。

MA 容错机制的目标在于避免故障产生、增强 MA 对故障的预测、防范及故障后的恢复能力。目前几种典型的容错途径包括：

(1)提高服务设施鲁棒性：MA 任务求解过程链中最易遭到破坏的环节是服务设施，提高服务设施的鲁棒性能够有效地避免故障产生。一般采用为 MA 分配单独的地址空间、维持 MA 备份的方式提高服务设施的容错性，如 Safe-TCL[1,2]。但过程的"灾难共享"特性决定了某一环节的改善不能有效地提高整个求解过程的容错性能。

(2)完善 MA 的安全机制：MA 自身的安全机制能够预测、防范服务设施施加的破坏性行为，保护自身数据的安全性。

(3)冗余任务求解：创建多个 MA 独立执行相同的任务求解，并在任务结束后比较运行结果。这种方法在保证 MA 结果有效性方面具有优势，但耗用大量网络资源，同时理论上并不提供可靠的容错性能。

(4)集中式冗余服务：特定主机作为冗余服务器，保存 MA 原始备份并跟踪 MA 的任务求解过程，若 MA 失效则通过重发原始备份提供故障恢复。这种依赖于特定结点的容错方法本身缺乏容错性，而且中间结果难以利用。

(5)分布式冗余服务：MA 容错性责任被分配到网络中多个非固定的节点承担，这些节点由冗余分配策略决定。这种方式能够弥补以上两种方法的弱点，提供较理想的容错性能。

3 移动窗口机制

本文提出的窗口容错机制可归入分布式冗余服务类型，其特点是在创建节点保存 MA 的永久原始备份，并在 MA 移动过程中保存最近的预定数量的 MA 备份，当 MA 成功地转移到新节点后，最老的备份被丢弃。这个动态的容错节点的分配策略类似一个固定长度的窗口随着 MA 的移动而移动，因此称之为基于移动窗口的容错机制。

3.1 相关定义

$V=\{v_1,v_2,\cdots,v_n\}$：节点的集合。

$E=V\times V\rightarrow N$：节点间的可达关系及权值。权值为 0 表示不可达，若节点间多个节点可达，权值越大则优先权越低。

$u=[v_1,v_2,\cdots,v_i,\cdots,v_n]v_i\in V$：MA 移动路径。若$\forall\ i,j,v_i,v_j\in u,v_i\neq v_j$ 则路径构成链，若 $v_1\neq v_n$ 则路径构成圈，MA 返回创建节点。

在移动路径 $u=[v_1,v_2,\cdots,v_n]$中，V_{i-1}称为 V_i 的直接前趋节点，V_{i+1}称为 V_i 的直接后继节点，$[v_1,v_2,\cdots,v_{i-1}]$和$[v_{i+1},V_{i+2},\cdots,V_n]$分别称为 V_i 的前趋结点和后继结点，$\mathrm{Pre}_j(V_i)$表示 V_i 节点的第 j 个前趋。

当只考虑 MA 容错性相关属性时，可将 MA 用以下元组表示：$MA(H,L,O,P,T,W)$，其中：H,L,O,P,T,W 可以作为 MA 的内部状态。H(Host)表示 MA 所在主机名；L(Length)表示移动窗口的长度，此长度决定了容错性的强弱；O(Order)表示此节点在窗口中的次序，O 越大，MA 所担负的容错责任越大；P(Path)记录在此节点前访问过的节点并保持访问次序，如果 $H=V_i$，则 $P=[v_1,v_2,\cdots,v_{i-1}]$；$T$ 表示 MA 允许在一个节点停留的时间；$W=T*(L-O)$决定了移动窗口内节点的等待时间，若在 W 时间后仍未得到后继节点的消息，即认为 MA 在后继节点崩溃。

移动窗口的上界是 MA 当前正在活动的节点，称为活动 MA；移动窗口内的其他 MA 称为备份 MA，不具有活动能力。正常状态下的移动窗口 $W=[\mathrm{Pre}_{i-L+1}(V_i),\cdots,V_i]$，其中 V_i 是活动节点，L 是移动窗口的长度，即保存 MA 备份的节点数。

MA 存储设施(MA Repository)是每个 MA 服务设施应该具备的 MA 存储空间，当服务设施所在主机处于 MA 所定义的移动窗口中时，MA 连同任务求解中间结果就保存在 MA 存储设施中；当 MA 被移出窗口之外时，它在 MA 存储设施中也同时被删除。

3.2 移动窗口工作原理

当 MA 被创建、发送后，MA 的创建节点将在其 MA 存储设施中保留一个永久备份，承担起集中冗余责任，它的作用是保证在最坏情况下恢复 MA。MA 创建者在 MA 的移动窗口中永远处于下界，是窗口中的静态结点，而在下面的移动窗口表示中缺省只列出动态结点。

首先考虑移动窗口工作过程中的一般情况。假设当前移动窗口 $W=[V_{i-l+1},\cdots,V_i]$，则窗口内节点对应保存的 MA 为(为简化略去创建结点)：

$$W_{ma}=\begin{bmatrix}MA(V_{i-l+1},L,1,[v_1,\cdots,v_{i-1}],T,T*(L-1)),\\ \cdots,MA(V_i,L,L,[v_1,\cdots,v_{i-1}],T,0)\end{bmatrix}$$

当在 V_i 节点完成任务求解后，根据可达关系 E 确定具有最小权值的后继访问节点 V_{i+1}，V_i 节点的

服务设施建立与 V_{i+1} 节点服务设施的连结，将 MA 传送到 V_{i+1} 节点。如果 V_{i+1} 节点接收此 MA 成功，将 $MA(V_{i+1},L,L,[v_1,\cdots,v_i],T,0)$ 作为新的 MA 插入 MA 存储设施，并向移动窗口中的节点发送 RECEIVE_OK 原语，移动窗口内节点是 P 中的最后 L 个节点。

如果 V_i 节点接收不到 RECEIVE_OK 原语，则认为出现了传输错误，建立连接进行重发。如果重复三次仍然不能够接收到 RECEIVE_OK 原语，根据 E 重新确定 V_{i+1}。重复此过程直到从 V_{i+1} 接收到 RECEIVE_OK 原语为止。如果无可用的 V_{i+1}，则将 MA 回溯至直接前趋。这个过程实现了传输过程的容错。

移动窗口内节点接收到 RECEIVE_OK 原语后，认为 MA 已成功地移动到下一个节点，于是将自身在移动窗口内的位置减 1，即 $O:=O-1$，然后将等待时间 $W=T*(L-O)$ 根据新的 O 值进行重新设置。如果 $O=0$，则节点被移出移动窗口，将 MA 从自身的 MA 存储设施中删除。

$W=[V_{i-l+2},\cdots,V_{i+1}]$ 为转移后移动窗口。随着 MA 在网络中的移动，移动窗口也按照这样的规律向前推移，直到 MA 到达任务求解终点，即 $W=[V_{n-l+1},\cdots,V_n]$。

到达终点 V_n 后，V_n 除了向移动窗口内的节点发送 RECEIVE_OK 原语外，还发送一个 DESTINATION 原语，DESTINATION 原语的语义为撤销移动窗口。并通知移动窗口中的其他节点将其从 MA 存储设施中删除。

移动窗口中创建结点是一个特殊的结点，它在窗口中的次序永远为零，因此永远处于窗口的下界。当收到活动节点的 RECEIVE_OK 原语后，它重新设置等待时间 $W=T*L$，并不改变窗口次序值。

窗口长度 L 决定容错性能。如果 L 取值合理，那么移动窗口能够提供足够的容错保证。创建结点的容错责任极小，这意味着可以充分保证创建结点的异步性，这点对于流动用户尤为重要。当创建结点重新打开(或由于崩溃而恢复)后，它仅仅重设等待时间 $W=T*L$，而不需要做其他工作。

3.3 故障恢复

基于窗口的容错机制能够有效地恢复由网络传输和服务设施故障造成的 MA 崩溃。3.2 节中介绍了网络传输故障的恢复机制，它主要通过 MA 转移方和接受方的协商实现。下面主要阐述服务设施故障或智能体故障的恢复机制。

移动窗口内的 L 个节点中保留有 L 个 MA，其中只有最后一个 MA 是活动 MA，前面 $L-1$ 个节点是备份节点，它们在窗口中的位置越高，担负的容错性责任越高。备份节点的容错责任是由等待时间决定的，等待时间 $W=T*(L-O)$。

假设当前活动节点是 V_i 节点，那么 V_{i-1} 节点是容错性责任最高的备份节点，它的等待时间 $W=T$，即为创建者确定的在一个站点停留的最大时间，如果超过了这个时间仍未收到 RECEIVE_OK 原语，即认为 V_{i-1} 的直接后继节点遭到破坏。

如果当前活动结点由于特殊原因需要停留更长时间，那么它必须以 T 为时间周期，向移动窗口内的其他节点发送 MA_ACTIVE 原语使其他节点等待时间重新开始计数。

当一个备份节点在等待时间内仍然没有收到 RECEIVE_OK 原语，即开始行使容错职责。它向移动窗口中位置靠后的备份节点发送 MA_LOST($L-O$)原语，将自己在窗口内的位置置为 L，成为活动节点；窗口内其后的备份节点在收到 MA_LOST($L-O$)原语后，将原语的参数($L-O$)累加到自己的窗口次序值形成在窗口中新的位置，重新设定等待时间。重新排列窗口位置后，由当前节点确定下一节点，重新发送 MA。此时窗口将是不完整的，在经过几次正常的移动之后，将重新建立一个完整的窗口。如果在一个结点连续失败三次，则不再将其做为 MA 的路由结点。

当崩溃的系统重新恢复执行后，由于其中的所有 MA 都存在失效可能性，因此必须对所有当前 MA 存储设施中的 MA 进行验证。具体验证过程通过查询前趋节点完成。对于活动 MA，向其直接前趋节点发出 GET_ORDER 原语，如果得到的位置值为 $L-1$，证明此活动 MA 仍为有效的，于是它向移动窗口内其他备份节点发出 MA_ACTIVE 原语实施再同步；如果得到的位置值小于 $L-1$，则此活动 MA 已经失效，将其从 MA 存储设施中删除；对于备份的 MA，可能采用相似方法与其前趋和后继节点联系，如果能够重新保持同步则留在移动窗口中，若不能保持同步则将其从系统中删除。简单而有效的方法是假设其失效而将其直接删除。

当创建节点在等待时间 W 过程中不能收到活动节点的 RECEIVE_OK 原语，即认为所有后继全部崩溃，将重发 MA。如果创建结点的重发过程重复两次以上，应该对参数 L 进行调整。

3.4 存在的问题及对策

网络环境(尤其是广域网)的复杂性使窗口在移动过程中会遇到各种难以预料的问题,这些问题可能破坏移动窗口内节点数据的一致性,影响系统的正常工作,甚至造成窗口的紊乱。这些可能产生的情况包括:

(1)节点间同步:移动窗口机制能够正常工作的前提条件是保持窗口内节点的同步,窗口内所有节点基于MA携带的"时间戳"(由于节点存在的差异,直接采用各节点时钟是不准确的)建立一个统一的虚拟时钟,由此虚拟时钟提供同步。但移动窗口中节点的分布特征及网络环境的复杂性会产生消息包丢失或网络延迟现象使同步信息不能够同时到达后备节点,这在严重的情况下会造成移动窗口内的节点间失去同步,导致在系统完全正常的情况下激活后备节点或容错责任低的节点先于容错节点高的节点被激活。严格控制节点间的同步是可能的,但将大大增加节点间的通讯量和系统的复杂程度。

(2)*MA*存储设施资源紧张:网络中主机的磁盘资源都是比较有限的,如果许多*MA*都定义较大的窗口长度值,那么将会占用许多备份节点的磁盘空间。在*MA*存储设施空间有限而又必须接纳活动*MA*时,必须基于某种策略强制刷新*MA*存储。显而易见,备份*MA*比活动*MA*在策略中具有更高的优先级。

此外,容错责任应做为制订刷新策略的重要因素。*MA*中$L-O$值越大,则其处于自己移动窗口的位置越靠后,容错责任越小。系统将首先删除$L-O$值最大的备份*MA*,然后考虑$L-O$值较小的备份*MA*。如果被删除的*MA*处于自己窗口的中间位置,那么应该考虑向其窗口位置值低的节点发送MA_LOST原语,使它们调整位置值和等待时间,以加速新的完整移动窗口的形成过程;如果不发送MA_LOST原语的话,建立一个完整的移动窗口需要耗费更长的时间。

4 性能及评价

采用移动窗口容错机制具有以下特点:

(1)窗口长度决定容错性能:假设*MA*在移动窗口内L个节点发生崩溃的概率分别为P_{i-L+1},…,P_i,并且这些事件都是独立事件,那么采用移动窗口机制后,*MA*崩溃的概率$P(W)=P_{i-L+1}*\cdots*P_i$,由此可见,窗口长度$L$越大,在网络中保存的*MA*备份越多,容错性能越好,耗费的资源也越多;反之亦相反。当$L=1$时,*MA*不在网络中保留备份,容错性能最低,但也不需要额外的通讯和存储代价。窗口长度L可根据网络及主机容错质量进行设置。

(2)分布式冗余分配策略:担负容错责任的站点是在*MA*任务求解过程中动态定义的,除非在最坏的情况下(此时由创建节点重发),不需要集中式的冗余服务,保证系统的异步性。

(3)充分保护*MA*任务求解的中间结果:移动窗口中节点位置越高,担负的容错性责任越大,同时它具有的中间结果也越多。这样就尽可能地利用了*MA*前的任务求解过程的中间结果,而不是恢复*MA*的原始状态,重新开始一个新的任务求解过程。

由于网络环境的复杂性,如何有效地维护移动窗口内节点的同步关系是系统的一个重要问题和难题,也是移动窗口机制今后急需解决的重点。

结论 移动窗口的容错机制将*MA*的容错性责任分布到任务求解过程中的不同节点,能够通过定义窗口长度改变容错指标来适应不同质量网络环境下容错性的要求,在提供参数可调的容错性能的同时,不会严重影响每个节点的服务效率,是一种可操作的*MA*容错机制,并已在本单位的基于智能体分布式构件管理中应用。

参考文献

1 Gray R S. Agent TCL: A transportable agent system. Dartmouth College, 1995

2 Gray R S. Agent TCL: A flexible and secure mobile system. Dartmouth College, 1995

3 IBM. Mobile Agent Facility Specification, OMG TC Document. August 1996

4 OMG. The Common Object Request Broker: Architecture and Specification. July 1995