维普资讯 http://www.cqvip.com

计算机科学 2002Vol. 29№. 10

基于移动 Agent 的新型分布式文件系统研究*)

The Research of Mobile Agent-based Distributed File System

卢 军 卢显良 韩 宏 许 腾
(电子科技大学计算机学院 成都 610054)

Abstract In general, conventional distributed file systems is designed for LAN environment. They always play worse performance in WAN than in LAN. In this paper we present a new distributed file system - The Mobile Agent-based File System (MADFS). The intent of MADFS is to minimize the latency overhead inherent to the distribution of a large file system in WAN. MADFS organizes hosts into a hierarchical structure, uses Mobile Agents as the underlying transfer, communication and synchronization mechanism and uses hierarchical-and-converge cache coherency protocol to minimize network usage. MADFS can achieve better performance and availability than conventional distributed file system in WAN. In this paper, the architecture of MADFS, performance analysis, cache coherency protocol and security are discussed. At last of this paper, the disadvantage and future work of MADFS are presented.

Keywords Mobile agent, Distributed file system, Performance

1. 引言

现代分布式文件系统的发展趋势[1]是 Transparency、Scalability、Unix Semantics、Reliability、Adaptation 和 Security。随着技术的发展，人们希望建立一种分布式文件系统来管理更广范围的文件资源，例如管理 WAN 环境中的文件资源，甚至管理 Web 环境中的文件资源。传统分布式文件系统大多是针对高带宽、低时延的 LAN 设计的，不能适应低带宽、高时延的 WAN 环境[2]，主要体现在：(1)文件存储协议、Cache 管理机制不适合 WAN。传统分布式文件系统的文件传输协议、Cache 管理机制是针对高带宽、低时延的 LAN 设计的，在低带宽、高时延的 WAN 下性能很差。(2)伸缩性差。传统的单一中央文件服务器模型完全不适应 WAN 环境下的分布式文件系统，因为 WAN 环境下的分布式文件系统可能有成千上万个客户，巨大的文件数量和存储规模将使中央服务器不堪重负。(3)可用性差。WAN 中的分布式文件系统将在更广范围内容纳大量的服务器和主机，在这种情况下一台服务器或主机崩溃或者网络连接中断的可能性更大，因此新的分布式文件系统必须可以容忍一台服务器或主机暂时不可用或者离线。

Sprite[3]是用 RISC 进行符号处理课题组开发的分布式文件系统，Sprite 和 NFS 一样是基于 Remote Procedure Call (RPC)来实现的。RPC 是针对 LAN 设计的，在 WAN 中性能很差[4]，导致 Sprite 很难在 WAN 中应用。Coda[5]是卡同基梅隆大学开发的分布式文件系统，Coda 通过服务器执行 Call-back 来实现 Cache 一致性管理。虽然 Coda 采用了 AVSG (Access Volume Storage Group)机制来提高 Call-back 的执行效率，但是由于 Coda 的体系结构是单层的 Client/Server 结构，所有的 Call-back 工作必须由少数文件服务器担任，因此随着系统规模的增加文件服务器将成为系统性能的瓶颈。

本文提出了一种基于移动 Agent 的新型分布式文件系统—MADFS。MADFS 使用分层体系结构，将整个文件系统划分为若干域。每个域由域服务器管理，所有域服务器由主服务器管理。这种体系结构避免了少数中央服务器成为系统扩展的瓶颈。MADFS 使用移动 Agent 来实现数据传输、通信和同步机制，比使用 RPC 更适应在 WAN 环境中工作。MADFS 采用了高效的 Agent 代码加载机制和"分层汇聚"Cache 管理机制来降低在 WAN 中的网络开销。上述特色使 MADFS 拥有很好的伸缩性、适应性和可用性，比传统分布式文件系统更适应在 WAN 环境下运行。

本文介绍 MADFS 的体系结构之后介绍其高效 Agent 代码加载机制以及其中的 Cache 管理机制，并对其安全问题进行了分析，最后提出了未来的工作。

2. MADFS 体系结构

传统的分布式文件系统，例如 NFS、Coda、Sprite 采用一层的 Client/Server 模型。这种模型在 LAN 中工作得很好，但却不适合在 WAN 中工作，因为单层的 Client/Server 模型要

*)本文受国家 95 重点攻关项目支持。卢 军 博士生。卢显良 教授，博士生导师。韩 宏 博士生。许 腾 讲师。

技术的研究，并将研究的初步成果用于 RoboCup 仿真系统中，已取得很好的效果。

参 考 文 献

1 Tambe M. Towards Flexible Teamwork. Journal of Artificial Intelligence Research, 1997, 7: 83～124
2 Tambe M. Executing Team Plans in Dynamic, Multi-agent Domains. In: Proc. of the AAAI Fall Symposium on Plan Execution: Problems and Issues, 1994
3 Marsella S, et al. On being a teammate: Experiences acquired in the design of robocup teams. In: Proc. of the 3rd Intl. conf. on autonomous agents, Agents'99
4 Marcus J H, et al. Dynamic Environment: Autonomous Netrek Agents. In: Proc. of the First Intl. Conf. on Autonomous Agents (AA), Marina del Rey, CA, Feb. 1997. 332～339
5 Ch'ng S, Padgham L. From Roles to Teamwork: a framework and architecture, (1997) in Applied artificial Intelligence, special issue on Robocup, 1997
6 Cohen P R, Levesque H J. Confirmation and joint action. In: Proc. of the Intl. Joint Conf. on Artificial Intelligence, 1991a
7 Jennings N. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. Artificial Intelligence, 1995, 75
8 Jennings N. Commitments and conventions: the foundation of coordination in multi-agent systems. The Knowledge Engineering Review, 1994, 8
9 Grosz B, Kraus S. Collaborative plans for complex group actions. Artificial Intelligence, 1996, 86: 269～358

维普资讯 http://www.cqvip.com

求在网络中进行大量的通信，这些通信消耗不适应 WAN 环境。

MADFS 的体系结构如图 1 所示，MADFS 由若干 Domain（域）组成，域内通过高速 LAN 连接，域间是低速 WAN 连接。每个域由若干服务器和主机组成。在 MADFS 中，一个域担任 MADFS 的主域，负责协调和管理 MADFS 的运行。MADFS 中每台服务器和主机都运行 Agent 支持环境，整个 MADFS 构成一个大的移动 Agent 运行系统。

在 MADFS 中，Agent 担负着数据传输、通信、管理协作等任务。如图 2 所示，在 MADFS 中 Agent 可以划分为以下几类：(1)IA(Interface Agent)：IA 运行在客户机上，接受客户机的文件调用，然后通过派遣、控制或和其他 Agent 协作来进行工作；(2)WA(Working Agent)：WA 接收来自 IA 的命令或者申请，在网络中移动并执行文件操作，然后返回执行结果信息；(3)DMA(Domain Manage Agent)：负责 MADFS 一个域中名字管理、Cache 管理、属性管理、空间管理和访问控制管理。DMA 可以自身复制和移动以靠近使用的数据来获得更好的效率和性能；(4)MMA(Main Manage Agent)：MMA 担负着协调与管理 MADFS 中所有 DMA 运行的任务。MMA 和 DMA 相互协作，完成整个 MADFS 中的各项管理工作。

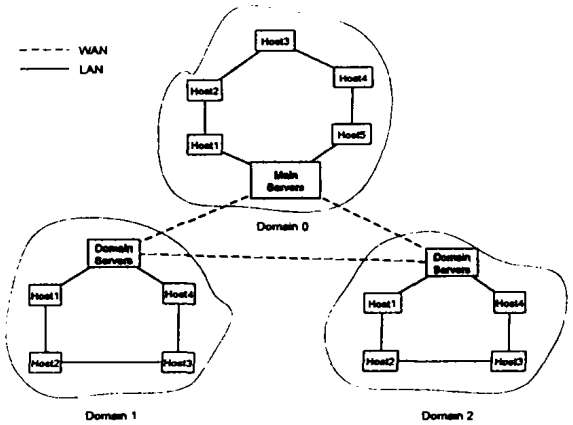

图 1　MADFS 体系结构图

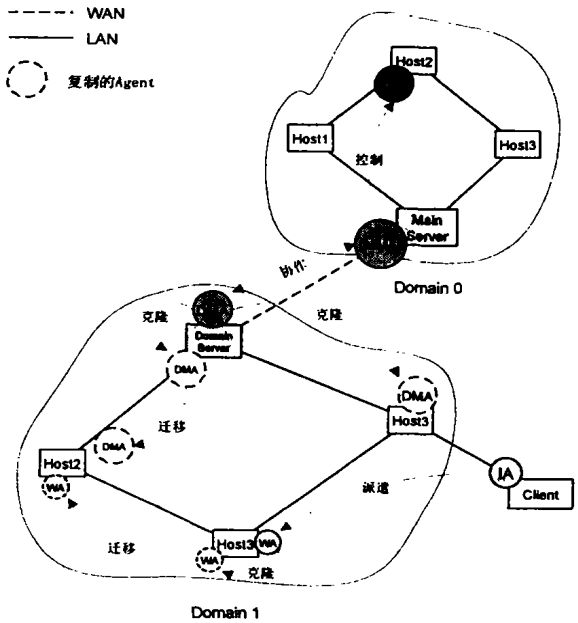

图 2　MADFS 中的 Agent 分工协作图

MADFS 分层体系结构的优点是：(1)DMA 负责管理一个域，MMA 负责管理所有 DMA。这种体系结构避免了 MADFS 中所有管理工作集中在少数服务器上，而使这些服务器成为系统扩展的瓶颈；(2)在 MADFS 中域内通过 LAN 连接，带宽大、延迟小。因此 MADFS 在域内通信时可以使用针对 LAN 设计的协议来获得较高的性能；同时在域内还可以认为主机之间是相互信任的，可以适当地降低安全标准来获得更好的性能。当跨域通信时，可以使用针对 WAN 环境设计的通信协议、Cache 管理协议和安全协议，以获得较好的性能和安全性；(3)文[2]表明在一个局域网中，一个主机的大约 2/3 需要读的文件都已经被本网其他主机读取了。文[6]也表明，在一个局域网中多个主机的 Cache 协作可以有效地提高 Cache 命中率。本文将在第 4 节详细叙述 MADFS 中使用的基于移动 Agent 的高效 Cache 管理机制。这种高效 Cache 管理机制的基础就是分层体系结构。

3. MADFS 中的 Agent 代码加载机制

传统分布式文件系统是通过 RPC 机制来进行通信的。NFS 是典型的使用 RPC 机制的分布式文件系统。研究表明，RPC 在高带宽、低时延的局域网中工作得很好，而在低带宽、高时延的 WAN 中性能很差[4]。为了避免 RPC 在 WAN 环境中的低性能，MADFS 使用移动 Agent 来实现数据传输、通信和同步机制。

文[7]通过对三种移动 Agent 平台的试验分析得出，移动 Agent 在目的机上的安装时间和 Agent 的代码加载机制有很大的联系。如果 Agent 全部所需运行代码已经包含在 Agent 中，安装时间将非常小，但 Agent 在网络中移动所带来的通信量将较大；如果 Agent 中仅仅包含部分运行代码，Agent 运行时就必须从代码服务器加载所需代码，导致安装时间较长，但 Agent 在网络中移动时通信量将较小。一种可行的方案是将移动 Agent 的代码服务器设置在每个域中，当 Agent 需要加载代码时，可以从本域中的代码服务器加载代码。

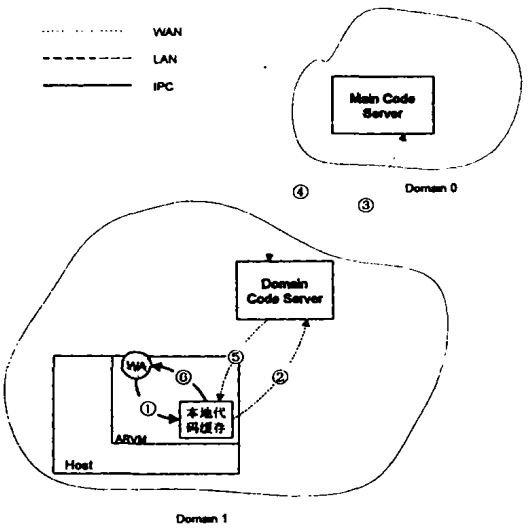

图 3　Agent 代码加载示意图

在 MADFS 中有一个 MCS(Main Code Server)，每个域中有一个 DCS(Domain Code Server)。MCS 中存放着 Agent 运行所需的全部代码，DCS 可以从 MCS 获得并缓存代码，这些缓存的代码可以供本域内的其他 Agent 使用。除了 MCS 和 DCS 上存放 Agent 运行所需代码以外，每台主机上也有代

维普资讯 http://www.cqvip.com

码的本地缓存可以提供给在其上运行的 Agent 高速加载代码时使用。如图 3 所示，在最差的情况下，Agent 所需的运行代码在 Agent 所在主机的本地代码缓存和 DCS 中都没有，必须从 MCS 中加载，Agent 的代码加载过程要经过 6 步：(1)Agent 向主机本地代码缓存查询代码；(2)Agent 所在主机向 DCS 查询代码；(3)DCS 向 MCS 查询代码；(4)MCS 向 DCS 返回要求的代码；(5)DCS 向 Agent 所在主机返回代码，Agent 所在主机将代码缓存；(6)Agent 所在主机向 Agent 提供所需代码。其中(3)、(4)两步是在 WAN 中通信，速度较慢；(2)、(5)是在 LAN 中通信，速度较快；(1)、(6)两步是在 Agent 所在主机上通过 IPC(InterProcess Communication)来实现的，速度最快。由于主机将获得的代码都放入本地代码缓存中供以后 Agent 加载时使用，同时 DCS 从 MCS 取回的代码也将在 DCS 中进行缓存供本域中的其他主机使用，因此在大多数情况下 Agent 加载代码时，只需经过(1)、(6)两步(本地加载)，或者经过(1)、(2)、(5)、(6)四步(本域加载)，具有很快的速度。文[7]中通过对三种移动 Agent 系统的比较分析得出，当在局域网中加载代码时，Agent 的加载速度非常快，远远小于 Agent 在 WAN 中移动所需要的时间。

从上述可以看到，在 MADFS 中 MCS、DCS 和主机本地代码缓存构成了一个分级动态代码加载体系，这种体系使 Agent 在加载代码时可以尽量从本地或本域加载，具有很快的速度，同时避免了在 WAN 上重复的通信消耗。

4. MADFS 中的 Cache 管理机制

MADFS 是在 WAN 上使用的分布式文件系统，因此 MADFS 的设计目标是既提供 Unix 语义，也应该尽量减小网络通信量，特别是在 WAN 上的通信量。为此，我们为 MADFS 的 Cache 管理设计了“分层汇聚锁”机制。这种机制可以有效降低服务器负荷，提供 Unix 语义支持，具有较小的网络通信量，较好地适应 WAN 的运行环境。

在 MADFS 中每一个文件都有“Reading Lock(读锁)”和“Writing Lock(写锁)”，只有获得了读锁或者写锁才能对文件进行读或写。一个拥有读锁的客户可以在本地缓存文件，拥有写锁的客户可以独享地在本地缓存中对文件进行任意读写操作而不需要和服务器交互。服务器既可以向客户分发读锁和写锁，也可以要求客户释放读锁和写锁。当一个客户释放读锁时，必须使自己的读缓存无效；当客户释放写锁时，必须将自己本地缓存中的脏数据刷新到服务器。

MADFS 中有一个唯一的 MCS(Main Consistence Server)，每个域有一个 DCS(Domain Consistence Server)，MCS 和 DCS 相互协作完成 MADFS 中的 Cache 管理。MADFS 的文件锁管理分成两层，DCS 负责所在域的文件锁管理，而 MCS 负责所有 DCS 的文件锁管理。如图 4 所示，当一个域中某台主机申请一个文件的锁时，该主机首先向所在域的 DCS 申请锁。DCS 收到锁请求后再向 MCS 申请锁，MCS 收到请求后向 DCS 颁发锁，DCS 再把获得的锁颁发给申请该锁的主机。

在上述分层锁申请过程中，DCS 的一项重要工作就是“汇聚”本域内的锁请求，从而减少 WAN 上的通信量，提高 Cache 管理效率。每个 DCS 都拥有一张锁记录表格，在表格中记录了本域中主机拥有读写锁的情况。假设在某个域中有一个主机 A 申请文件 file.dat 的读锁，主机 A 首先向 DCS 发出锁申请，DCS 在自己的锁记录表格中查找是否已经有主机拥有了 file.dat 的读锁，这时有两种情况：(1)DCS 发现本域还没有任何主机拥有 file.dat 的读锁，那么 DCS 将向 MCS 发出请求申请 file.dat 的读锁，当 MCS 向 DCS 颁发 file.dat 的读锁后，DCS 将锁颁发给主机 A，然后 DCS 在自己的锁记录表格中记录下主机 A 获得文件 file.dat 的读锁。当主机 A 获得 file.dat 的读锁后，从远端主机取回 file.dat 文件的内容，缓存在主机 A 上。(2)DCS 发现在本域中已有主机 B 拥有 file.dat 文件的读锁。这时 DCS 不需再次向 MCS 申请 file.dat 的读锁了，而只需将 file.dat 的读锁颁发给主机 A，然后在锁记录表格中添加主机 A 拥有 file.dat 的读锁记录即可。当主机 A 拥有 file.dat 的读锁后，不需要从 file.dat 文件所在的远端主机上取回 file.dat 文件的内容，而可以直接拷贝主机 B 的 file.dat 文件缓存即可。从上述过程中我们看到，DCS 可以“汇聚”一个域中的读锁请求，当一个域中若干主机先后请求同一个文件的读锁时，只在第一次读锁请求时需要向 MCS 进行远程锁请求通信，同时也仅仅在第一次时需要远程文件内容传递，而从第二次开始，锁请求和文件内容的拷贝全在域内进行。这种“汇聚”机制具有如下优点：(1)有效地减少了 MADFS 中管理文件读写锁和复制文件内容的通信量，特别是在 WAN 上的通信量；(2)MCS 中不需维护每一个主机的文件读写锁状态，而仅仅需要维护 DCS 拥有读写锁的状态，这样大大降低了 MCS 维护文件读写锁的开销。

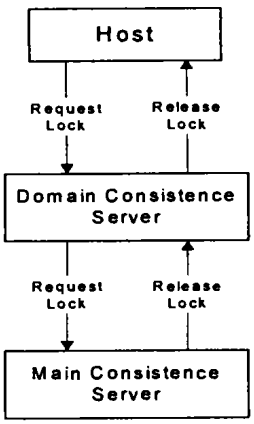

图 4　MADFS 中两级锁申请/释放的机制

MADFS 的分层 Cache 管理机制在具有上述优点的同时，还充分利用了移动 Agent 的特点来提供更好的性能。如图 5 所示，设 Host5、Host6、Host7 拥有一个文件(假定是 file.dat)的读锁，Host3 申请 file.dat 的写锁，整个的锁申请和释放过程包括如下的步骤：(1)Host3 生成一个 Agent，发送到 DCS1 上申请 file.dat 的写锁；(2)Agent 到达 DCS1 后，查找 DCS1 的锁记录表，发现本域内没有其他主机拥有 file.dat 写锁，于是 Agent 移动到 MCS 上，向 MCS 申请 file.dat 的写锁；(3)Agent 到达 MCS 后，查找 MCS 的锁记录表，发现 DCS1 和 DCS2 拥有 file.dat 读锁。依据 Multiple-Reader/Single-Writer Locks(多读/单写锁)Cache 一致性协议，在颁发 file.dat 写锁给 DCS1 前，必须要求 DCS1 和 DCS2 释放读锁。Agent 自身复制变成两个 Agent(假定是 Agent1 和 Agent2)。Agent1 移动到 DCS1，Agent2 移动到 DCS2，如图 5 中的③和③所示；(4)当 Agent1 到达 DCS1 后，查找 DCS1 的锁记录表格，发现 Domain1 中有 Host5 拥有 file.dat 读锁，Agent1 向 Host5 发出释放读锁的请求，Host5 收到请求后，释放读锁。Agent1 得到 Host5 释放读锁的确认后，依次返回 DCS1 和 MCS。Agent2 在 Domain2 中进行同样的工作，得到 Host6 和 Host7 释放读锁的确认后，依次返回 DCS2 和 MCS；(5)Agen-

维普资讯 http://www.cqvip.com

1 和 Agent2 到达 MCS 后相互交换信息，Agent2 就结束运行。Agent1 在 MCS 的锁记录表中删除 DCS1 和 DCS2 拥有 file.dat 读锁的记录，然后添加 DCS1 拥有 file.dat 写锁的记录；(6)Agent1 移动到 DCS1 后，在 DCS1 的锁记录表中添加 Host3 拥有 file.dat 写锁的记录；(7)Agent1 返回 Host3，告知成功获得 file.dat 写锁，然后 Host3 取回 file.dat 的内容，在本地缓存，进行读写修改。从以上步骤我们看到，充分利用移动 Agent 的移动性、复制性、协作性，可以有效地降低文件读写锁管理中的通信量，降低 DCS 和 MCS 的负荷，使 MADFS 在 WAN 上获得良好的性能。

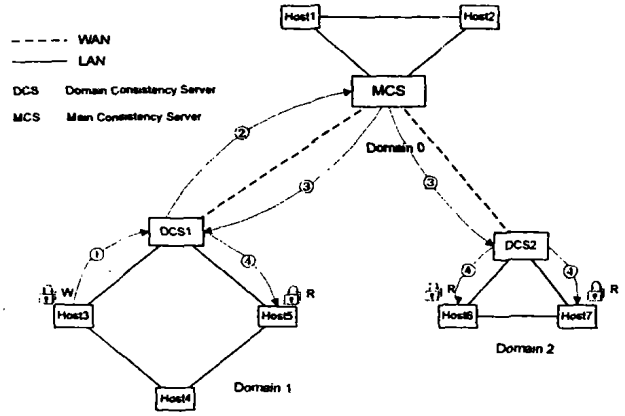

图 5 MADFS 的分层缓存体系

5. MADFS 的安全性

MADFS 是针对开放网络环境设计的，例如 WAN 和 Web。在这样的环境中很容易受到各种错综复杂因素的影响，因此必须建立一种完善的安全机制来确保 MADFS 的安全性。

MADFS 使用移动 Agent 来实现数据传输、通信和同步机制，因此 MADFS 的安全性和移动 Agent 的安全性密切相关。MADFS 的安全性主要体现在三个方面：(1)Agent 通信、传输、协作的安全性。Agent 之间通信不受干扰和更改；Agent 在移动过程中不被修改；Agent 运行不受恶意非法 Agent 的控制。(2)保证 MADFS 中服务器不受恶意或伪装 Agent 的非法访问。Agent 在移动到一个服务器后，服务器应该首先对 Agent 进行身份验证，证实该 Agent 是由允许访问的主机委托访问时才容许该 Agent 执行；其次，在 Agent 执行的过程中，服务器应该保证 Agent 只能执行预先设定的任务，而不能运行未设定的非法任务。(3)保护 Agent 不会在自己的非目的服务器上运行。在 MADFS 中，可以使用如下机制来提供安全保证：

1. **安全通信** 为了确保 Agent 之间通信的安全性，可以在 Agent 的通信层使用 SSL(Secure Socket Layer)协议来保证通信的安全。Agent 通信安全对于 MADFS 应该是透明的，SSL 能够很好地满足这一需求。文[8]中提出了一种基于"加密信道—权限控制"的安全机制，该机制使 Agent 之间的通信在加密信道上进行，并提供多层次的检查机制，可以满足 MADFS 中对 Agent 通信安全的保证。

2. **识别和认证** 在 MADFS 中，服务器只能给信赖的 Agent 以执行的权力，Agent 也只能在信赖的服务器上执行。因此 Agent 和服务器之间必须建立相互识别和验证机制。每个服务器都可拥有一些信赖主机的标识和密钥，凡是通过检验的 Agent 被认为是信赖的，赋予相应的权限，可以允许执行。每个 Agent 也拥有目的服务器的地址和密钥，在到达一个服务器时，不仅要接受服务器的身份验证，也要对服务器进行验证，确认服务器是否为可信赖的。文[9]中介绍了携带验证代码(Proof Carrying Code—PCC)技术，PCC 技术使服务器上执行的 Agent 具有由客户确定行为的安全特性。使用 PCC 可以阻止 Agent 执行不安全的代码。Agent 也可以使用数字签名来标记自己。使用数字签名可以验证 Agent 的真实性和完整性。在 MADFS 中，Agent 作为一个客户的代表访问服务器，Agent 可以使用该用户的签名作为其授权标志。

3. **访问控制** 在 MADFS 中所有的文件、目录都保护在 Access Control List(ACL)下。ACL 中记录了某种资源只能由某些客户访问，ACL 是由 Domain Manage Agent(DMA)来实现的。当客户需要访问一类资源的时候，客户派遣出的 Agent 就向 DMA 查询访问权限。DMA 通过验证该 Agent 的数字签名来判断该 Agent 是否具有访问资源的权限，如果有则允许访问，否则 DMA 将不容许对资源的访问。

6. 未来的工作

(1)在分布式文件系统中灾难恢复是一项非常重要的工作。对于使用几台中央服务器的分布式文件系统而言，一种简单有效的灾难恢复机制是使用备份的数据将系统所有服务器恢复到崩溃以前的状态。但是对于 MADFS 这样的方式很难实行，原因是 MADFS 是 WAN 环境下的分布式文件系统，服务器的地域分布是不集中的，很难将整个系统的状态一致地保存下来。因此研究一种简单易行的灾难恢复机制是 MADFS 未来一项重要的工作。

(2)在 MADFS 中，MCS 和 DCS 必须记录用户拥有文件读写锁的状态。如果 MCS、DCS 或主机崩溃，如何正确恢复文件读写锁拥有状态记录是一件复杂的工作，需要重点研究。

(3)使用移动 Agent 技术在降低网络通信量的同时也将增加服务器的负荷。对于可能有成千上万个 Agent 同时运行的系统中，系统中服务器的负荷将很大。所以研究 Light-Weight 型和高伸缩性的移动 Agent 系统将有助于构建大型 MADFS 系统。

结束语 本文提出了一种基于移动 Agent 的新型分布式文件系统 — MADFS。MADFS 使用分层体系结构，整个系统由若干域组成，域内通过 LAN 连接，域间通过 WAN 连接。每个域由域服务器管理，所有域服务器由主服务器管理。这种体系结构保证了 MADFS 具有很好的伸缩性和扩展性。MADFS 针对 WAN 设计了高效的分级 Agent 代码加载机制，既降低了网络通信量，又使 Agent 加载具有很快的速度。MADFS 在 Cache 管理中使用了"分层汇聚锁"机制，这种机制降低了服务器维护 Cache 一致性的负荷，网络通信量较小，非常适合在 WAN 中使用。此外，本文也对 MADFS 的安全问题和未来的工作进行了探讨。综上所述，MADFS 克服了传统分布式文件系统的不足，是一种适合在 WAN 中运行的新型分布式文件系统。

参 考 文 献

1 Wang R, Anderson T, Dalin M. Experience with a distributed file system implementation: [Technical report]. University of California, Berkeley, Computer Science Division, June 1997

(下转第 48 页)

维普资讯 http://www.cqvip.com

("kick-tree"),如果球在视力范围外队员应该采取其他行动("move-tree"),默认的状态下,当球不在视力范围内的时候是"找";在异构方法中,每一个队员有自己的算法,在机器人足球赛中,团队将具有22个算法(每个队员都有"kick-tree"和"move-tree"),Luke等人采取了伪进化(apseudo-heterogeneous)的方法来解决异构方法中耗时长的特点,把队员分组(squad),每组包含一个或者两个队员,每组内算法相同,这样可以大大节约时间。团队的适应度由竞争适应度(competitive fitness)方法来评价。

目前机器人足球赛中,进化团队协作的方法并不多见,1999年Jessica[19]曾经做过统计,在仿真机器人足球赛中采用进化计算方法的只有Darwin United和马里兰的GP Soccer Team。Darwin United也采用了进化计算方法,只是在具体实现上和马里兰的GP Soccer Team有很多不同之处[16,19]。进化团队协作的下一步发展目标是进化混合式团队结构,另外Jessica指出,完全由计算机进化的团队在近期还不能达到机器人足球队战胜人类足球冠军队这个理想,科学家们为达到该目标还需作坚持不懈的努力。我们共同期待五十年之后,这个宏伟目标能够得以实现。

结论 团队协作是多智能体系统研究的关键技术,因其应用前景广泛且具有较大的难度,引起了研究者们的广泛关注,近些年来,研究者在仿真机器人足球赛团队协作方面也取得了许多成绩,除了本文上述提及的几个团队外,仿真组还有其他一些比较有代表性的团队,比如PaSoTeam[20]采用刺激响应式结构,是基于行为的无通讯团队协作的代表。AIRG[21]把工业中全息(holonic)生产系统(HMS)中的全息模式引入到队员结构的设计中,建立了适合团队协作的混合式结构等等。但是,从目前来看要设计并实现一个健壮的团队协作系统,是一项艰巨而富有挑战性的课题,它依赖多种学科、多种技术的共同发展。同时如何提高整个团队的灵活性、容错性、适应性也是需要继续深入研究的问题,我们开展该领域的研究工作已有多年,目前正在设计和实现ND Soccer Team,有望在2002年底实现。衷心希望我们的工作能对今后团队协作的研究有所帮助。

参考文献

1 Yen J, Yin J. CAST: Collaborative Agents for Simulating Teamwork. In: Proc. of Intl. Joint Conf. on Artificial Intelligence, IJCAI-2001

2 Tambe M, Zhang Weixiong. Towards Flexible Teamwork in Persistent Teams. http: // www. isi. edu/teamcore/tambe/agent. html

3 Luke S, Spector L. Evolving Teamwork and Coordination with Genetic Programming. http: // helios. hampshire. edu/lspector/pubs/cooperation. pdf

4 Cohen P R, Levesque H J. Teamwork. Nous 25(4), Special Issue on Cognitive Science and Artificial Intelligence, 1991. 487～512

5 Tambe M. Implementing Agent Teams in Dynamic Muti-agent Environments. In: Proc. of AAAI Fall Symposium on plan Execution: problems and Issues, 1994

6 Tambe M. Towards Flexible Teamwork. Journal of Artificial Intelligence Research, 1997, 7: 83～124

7 Teamwork based approach for the OGI CoABS project. http: // www. cse. ogi. edu/CHCC/Agents/teamwork. html

8 http: // jmvidal. cse. sc. edu/papers/stonebook/

9 Veloso M, Stone P, Anticipation M B. A Key for Collaboration in a Team of Agents. http: // www. research. att. com/～pstone/papers. html

10 Kitano H, Tambe M, Stone P. The RoboCup Synthetic Agent Challenge'97. In: proc. of the Intl. Joint Conf. on Artificial Intelligence(IJCAI-97)Nagoya, Japan

11 Tambe M. Agent Architectures for Flexible, Practical Teamwork. In: Proc. of the 14th National Conf. on Artificial Intelligence, Providence, Rhode Island, 1997. 22～28

12 Grosz B J. Collaborative Plans for Complex Group Actions. Artificial Intelligence, 1996, 86: 269～358

13 Stone P, Veloso M. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. Artificial Intelligence, 1999, 110(2): 241～273

14 Marsella S, Tambe M. Experiences Acquired in the Design of RoboCup Teams: A Comparision of Two Fielded Teams Autonomous Agents and Multi-agent Systems, 4. 115～129

15 Tambe M, et al. ISIS: Using an Explicit Model of Teamwork in Robocup'97. http: // www-user. tu-chemnitz. de/～hla/soccer/resources/soccer_resources. html

16 Andre D, Teller A. Evolving Team. http: // www-2. cs. cmu. edu/～astro/astro-paper-page-frame. html

17 Luke S. Evolving SoccerBots: A Restrospective. http: // www. cs. umd. edu/～seanl

18 Luke S, Hohn C. Co-Evolving Soccer Softbot Team Coordination with Genetic Programming. In: Proc. of The First International Workshop on Robocup(IJCAI-97)

19 Lundberg J. Survey over Genetic Programming Approaches to RoboCup. http: // www. dsv. su. se/～ johank/courses/int4/robocup/papers/1999/int7/lundberg. pdf

20 Pagello E. Antonio D'Angelo Cooperative behaviors in multi-robot systems through implicit communication. Robotics and Autonomous Systems, 1999, 29: 65～77

21 Candea C, Hu Huosheng. Coodination in multi-agent RoboCup teams. Robotics and Autonomous Systems, 2001, 36: 67～86

22 http: // 202. 112. 108. 158/

23 Tambe M. Agent Architectures for flexible, practical teamwork. In: proc. of the National Conf. on Artificial Intelligence (AAAI), August

(上接第45页)

2 Blaze M, Alonso R. Toward Massive Distributed Systems. In: Proc. of the 3rd Workshop on Workstation Operating Systems, April 1992. 48～51

3 Ousterhout J, et al. The Sprite network operating system. IEEE Computer, 1988, 21(2): 23～36

4 Dube R, Rais C D, Tripathi S K. Improving NFS Performance over Wireless Links. IEEE Transactions on Computers, Feb. 1997

5 Satyanarayananan M, et al. A Highly Available File System for a Distributed Workstation. IEEE Transactions on Computers, 1990, 39(4)

6 Dahlin M, et al. A Quantitative Analysis of Cache Policies for Scalable Network File Systems. In SIGMETRICS, 1994

7 Ismail L, Hagimont D. A Performance Evaluation of the Mobile Agent Paradigm. ACM SIGPLAN Notices, 1999, 34(10): 306～313

8 Karjoth G, Lange D B, Oshima M. A Security Model for Aglets. IEEE Internet Computing, 1997, 1(4)

9 Necula G C, Lee P. Safe, untrusted agents using proof-carrying code. In Mobile Agent Security, volume 1419 of Lecture Notes in Computer Science. Springer-Verlag, 1998