计算机科学 1999Vol. 26№. 2

流动 agent：一种未来的分布计算模式

陶先平 吕 建 董 桓 李 新

（南京大学计算机软件新技术国家重点实验室 软件研究所 南京 210093）

摘 要 流动 agent 技术对于未来分布式系统的设计、实现和维护技术具有重要意义。它可有效地降低网络拥塞、克服网络隐患、其异步与自主运行特性提高了分布式系统的健壮性和容错能力。本文作者于 1998 年 7 月 20 日—7 月 24 日参加了在比利时布鲁塞尔举行的第十二届欧洲面向对象程序设计会议，会上，General Magic 公司的 D. B Lange 作了"Mobile Objects and Mobile Agents: The Future of Distributed Computing?"的特邀报告。作者在该报告的基础上，结合自身工作，完成此文，供同行参考，希望能够对国内在此方面的研究有所推动。

关键词 流动 agent，分布式计算，计算模式

1 流动 agent

什么是软件 agent？它由什么成分构成？它与一般程序有何不同？软件 agent 有何特性？本文从最终用户和运行系统两个角度给出以下定义：

从最终用户角度，agent 是一种程序，它代表用户，是用户实现其意图的软件助手，它因用户向它指派工作而起作用。

从系统角度，agent 是一个软件对象，生存于一个执行环境中并拥有以下基本特性：(1)反应能力。对环境变化的感知能力和应变能力；(2)自主性。执行动作的自我控制；(3)目标驱动。行为预知；(4)连续性。运行动作的持续性。除此以外，还可能拥有以下性质：(1)通信能力。可与其它 agent 进行通信；(2)流动能力。在主机站点之间转移；(3)学习能力。根据经验改进自身；(4)可靠性。对最终用户而言是可信任的。

并非所有 agent 都必须流动。agent 可以是静止不动的，以通常方式和周围环境交互，如各种形式的 RPC 及消息传递，这种不能或不需流动的 agent 称为静止 agent。

相反，流动 agent 不囿于它开始执行的系统，可在网络各主机间自由移动，在某个执行环境中被初建后，流动 agent 可携带自身状态和代码在网络中转移到另一环境中去，并可在该环境中恢复执行。其中"状态"是指 agent 在异地目标环境中恢复执行时所需的属性值，而"代码"是 agent 执行的必要条件。

据此，我们给出流动 agent 的定义：流动 agent 是一独立计算机程序，它可自主地在异构的网络上，按照一定的规程流动，寻找合适的计算资源、信息资源或软件资源，利用与这些资源同处一台主机或网络的优势，处理或使用这些资源，代表用户完成特定的任务。

2 流动 agent 的优点

流动 agent 技术给分布式系统的设计、实现和维护都带来了新的活力。

● 减轻网络负载：流动 agent 技术能较大地减轻网络上的原始数据的流量。分布式系统通常依赖于通信协议，这些协议在完成给定任务的过程中涉及多次交互行为，这将导致网络交通拥挤。如图 1 所示，流动 agent 使我们可以将一个会话过程打包，然后将其派遣到目的主机上去进行本地交互。此外，当进行远地主机的大量数据处理时，这些数据不应在网络上传来传去，而应在本地被处理完成。理由很

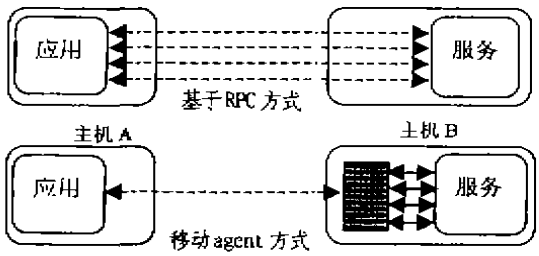

图 1 流动 agent 减轻网络负载

简单：应把计算移到数据上去进行，而不是把数据移到计算中来。

● 克服网络隐患：对那些重要的实时系统而言，如使用大规模工厂网络对进行加工制造的机器人进行控制的实时系统，系统需要对环境的变化作出实时的反应，但这种网络控制有很多的隐患，对实时系统而言是无法接受的。流动 agent 技术是一个很好的解决方法，因为 agent 可以从中央控制器被传送到各局部点激活，并在当地直接执行控制器的指令。

● 封装协议：当数据在分布式系统中进行交换时，每一台主机都有自己的网络协议，该协议将对传出数据进行编码，对传入数据进行解释。但是，协议经常为满足新的效率和安全需求而需要改进，而实现该协议的代码升级工作要么几乎不可能，要么相当困难，这样就会产生"遗产"协议；而流动 agent 能够直接流动到远地主机，建立起一个基于私有规程的数据传输通道。

● 流动 agent 异步自主运行：通常，流动设备上的计算皆依赖于昂贵而脆弱的网络连接，它要求在流动设备和固定网络之间建立持续的连接，这种要求从经济和技术的角度来讲都不易实现。但如图 2 所示，此种任务可以嵌入到流动 agent 中去，然后将它通过网络派遣出去。此后，流动 agent 就独立于生成它的进程，并可异步自主操作了；流动设备则可在稍后的时间里再连接并回收 agent。

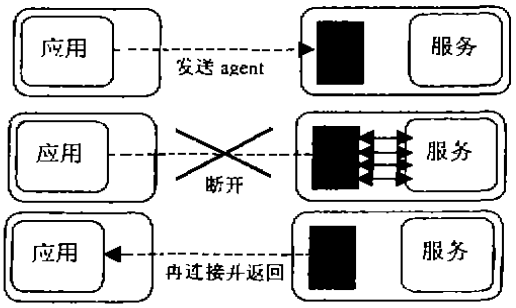

图 2　流动 agent 允许网络断开操作

● 流动 agent 的应变能力：流动 agent 具备感知其运行环境，并对环境变化作出反应的能力。多流动 agent 拥有在网络主机之间动态合理分布自身的独特能力，比如按一定规则来维持解决某个特定问题的最优配置。

● 具有自然异构性：网络计算平台往往是异构的。由于流动 agent 通常是独立于计算机和传输层，而仅仅依赖于其运行环境，所以流动 agent 提供了系统无缝集成的最优条件。

● 坚定性和容错性：流动 agent 具有对非预期状态和事件的应变能力，这使我们更容易创建坚定和容错性好的分布式系统。当关闭一台主机时，所有正在该主机上运行的 agent 会得到警告，并有足够的时间转移到另一台主机上并继续运行。

3　网络计算模式

流动 agent 为网络计算提供了一个强大的统一计算模式，它可以彻底改变分布式系统的设计和开发。以下我们将对 3 种典型的分布式计算模式进行概述和比较：客户/服务器模式，Code-on-demand（代码点用）模式和流动 agent 模式。

● 客户/服务器模式：如图 3 所示，在客户/服务器模式下，服务器对所能提供的资源（如数据库）服务进行广播，而实现服务的代码却驻留在本地服务器上。如果客户方对服务器上的某些信息资源感兴趣，它只需简单调用一个或多个服务器提供的服务即可。但客户方需要某些"智能"来决定该使用哪一个服务。这样，服务器拥有处理器资源、软件资源和信息资源。到目前为止，大多数分布式系统都是基于这种模式的。它们得到了广泛的技术支撑：RPC，CORBA 和 Java 的 RMI。

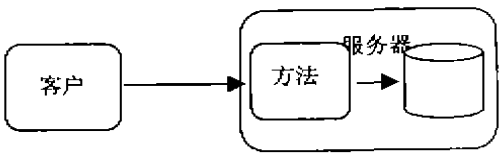

图 3　客户-服务器模式

● Code-on-demand 模式：如图 4 所示，按照该模式，我们在需要服务时，首先获得它的方法。也就是说，一台主机 A 最初由于没有代码而不能执行任务，但网络中另一台主机 B 可提供需要的代码。一旦 A 获得 B 中的代码，A 就同时拥有处理器和本地资源，计算就可在 A 中完成。与客户/服务器模式不同，A 无需知道远程主机的情况，因为可以下载所

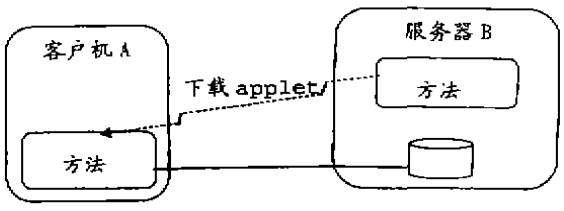

图 4　Code-on-demand 模式

需要的代码。Java 中的 applet 和 servlet 就是这种模式的极佳例子。applet 下载到浏览器中在本地运行，而 servlet 上载到远程 Web 服务器上运行。

● 流动 agent 模式：如图 5 所示，流动 agent 模式的关键特征就是网络中的任一主机都拥有处理资源、处理器和方法的任意组合的高度灵活性。方法（在流动 agent 的形式下）没有锁定在一台主机上，而是在整个网络内可共享。

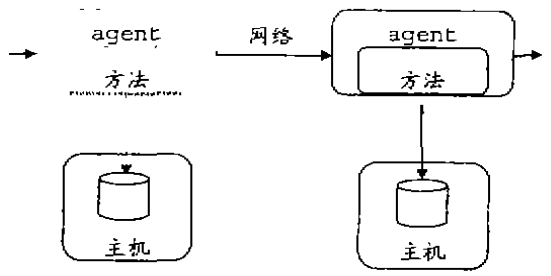

图 5　流动 agent 模式

比较这三种模式，可以发现计算模式的灵活性在不断增加。先是客户机和服务器融合为主机概念，然后是作为它们扩充的 applet 和 servlet 随着流动 agent 的出现而发生的融合及改进。

4　流动 agent 系统技术难点

● 克服计算环境的异构：流动 agent 是将整个 Internet 作为计算平台的，流动 agent 很可能要在不同的计算环境中自主地执行，因此必须首先解决流动 agent 的跨平台问题。

● 实现 agent 的自主流动：agent 的自主流动应该解决以下三个问题：1）agent 的流动规程。agent 流动的触发、目的地指定、agent 重执入口指定等 agent流动规程；2）agent 的通信模型。流动 agent 之间存在着协作关系，该模型应保证 agent 在流动时正确的通信和协作关系。3）agent 的迁移方式。流动 agent 在运行的过程中可能会因为本身的需要或意外事件暂停运行，需迁移到另外的站点上并继续执行等。

● 保证流动 agent 的安全性。安全性问题涉及三个方面：流动 agent 自身的安全保护、流动 agent 之间通讯的安全保护以及站点的安全保护。流动 agent的安全性是亟需解决但又较难解决的问题之一，它直接影响流动 agent 系统的实用性。

● 提供灵活方便的流动agent环境。流动 agent 系统用户需求是千变万化的，对应的流动 agent 也极其不同，因而用户必须可以根据自己的需要来开发合适的流动 agent。为用户提供一种方便的流动 agent编程语言以及相应的开发环境是流动 agent 系统所要解决的问题。

5　流动 Agent 的应用

电子商务：流动 agent 十分适合电子商务。商业事务需要对诸如股票行情等远程信息进行实时存取，需要 agent 之间的协商。不同的 agent 有不同的目标，为实现各自的目标采取不同的策略，agent 体现了各自创造者的意愿，代表他们运行和协作，流动 agent 技术在解决电子商务方面的问题上有着非常良好的前景。

分布信息查询：流动 agent 常常用于信息查询。把流动 agent 发往远方的信息源，直接进行查询再把结果发送回来，而不是将大量的数据取回来再作查询，减少了网络上传输的数据量。此外，流动 agent 也可在创建者计算机关闭时进行远程信息查询。

并行处理：由于流动 agent 能够在网络环境中生成多个副本，该技术的一个潜在应用是管理并行处理任务。如果计算需要大量的处理能力而必须分布到多个处理器上时，支持流动 agent 的主机构成的基础并行处理设施可以提供强大的支持。

个人助理：流动 agent 能够在远程主机上运行的能力使得它可以代表用户在网络中完成一些任务，成为人们的助手。它可以在远程主机上独自运行，不论网络是否连通，而发送者可以关掉自己的计算机。例如为了制订一个会议日程，用户只要发送一个流动 agent 和代表其他与会者的 agent 交互，这些流动 agent 经过协商后制订一个时间表。

信息发布：流动 agent 体现了 Internet 上的“推”模型。agent 可以发布诸如新闻和软件升级等信息，它可以带着新的软件组件和安装过程直接到达用户的个人计算机，进行自动升级和软件管理。

监视和通知：这是对流动 agent 异步特性的典型应用。agent 可以脱离自身的出生地，及时地监视某一信息源，等待该源上可用信息的出现。监视 agent具有比创建它们的进程更长的生命期，这一点是相当重要而且是必需的。

安全中介：流动 agent 一个有趣的现象是进行合作的参与者相互之间并不信任。这时，参与者可在互相认可的安全的主机上进行合作，而没有主机操纵流动 agent 的危险。

远程通信网络服务：对优质通信服务的支持和

管理主要体现在网络的动态配置和用户可定制。网络的物理大小和操作的严格限制需要流动 agent 技术来保证网络的灵活而有效。

工作流应用和群件：工作流的本质是支持信息在合作者间的流动。流动 agent 对此特别合适，因为除了流动性外，它还提供了工作流条目一定的自治性。每个工作流条目都可封装它所需要的行为和数据，然后在整个工作空间中流动，而不依赖于特定的应用。

6 目前的流动 agent 系统

Agent 系统的研制方兴未艾，本文只能提及几个基于 Java 的系统：Aglets、Odyssey、Concordia、Voyager 和 Mogent。

Aglets：采用 Java 中的 applet 模型，目的是引进 applet 的流动性。Aglet 一词本身就是由 agent 和 applet 组合而成。Aglets 设计简明，开发者希望程序员将会喜欢 Aglet 模型中从 applet 模型借鉴过来的许多特点。

Odyssey：General Magic 公司首先使用流动 agent 一词，并开发了第一个商业流动 agent 系统 Telescript。由于基于专有语言和网络体系结构，Telescript 昙花一现。针对 Internet 的普及和 Java 语言的巨大成功，General Magic 公司在基于 Java 的 Odyssey 系统中重新实现了流动 agent 系统。Odyssey 以 Java 类的方式有效地实现了 Telescript 中的概念，建立了一个 Java 类库，开发者可在此基础上建立自己的流动 agent 应用。

Concordia：日本三菱公司的 Concordia 是一个可进行流动 agent 应用的开发和管理的构架，它可延伸到任何支持 Java 的系统上。Concordia 由多个部件组成，这些部件由 Java 书写，它们组合起来构成一个完整的分布应用的计算环境。一个最简单的 Concordia 系统包含一个标准 Java VM，一个 Server 和一组 agents。

Voyager：ObjectSpace 的 Voyager 是一个基于 Java 并扩充了 agent 的分布计算平台。它增加了对象消息传递功能，同时也支持对象在网络上的移动。可以这样认为，Voyager 综合了基于 Java 的 ORB 特点和流动 agent 系统的特点。这样，Voyager 可以使 Java 程序员同时使用传统的和基于 agent 的两种分布式程序设计方法来开发网络应用。

Mogent：Mogent 系统是由南京大学计算机软件研究所自行研制的基于 Java 的流动 agent 系统，它包括开发环境、运行环境和监控环境三部分，具有流动性、协同性和安全性等特点。

值得一提的是，基于 Java 的流动 agent 系统有许多共同点，除了在编程语言上采用标准的 Java 虚拟机和对象序列化机制外，它们还采用了类似的基于 server 的软件体系结构，但在 agent 的移动及交互上的做法各有不同。

虽然现在大部分的流动 agent 系统都采用 Java 语言，但流动 agent 系统并不是非 Java 不可，Tcl 和 Python 语言也可用于实现流动 agent 系统。

Agent Tcl：一个用 Tcl 书写 agent 的流动 agent 系统，Darmouth 学院的 Agent Tcl 由导航和通信服务、安全保障机制以及调试和跟踪工具组成。Agent Tcl 的主要部分是一个运行于各个站点上的 server，该 server 支持一个 agent 的完整的执行状态（如局部变量、指令指针等）的迁移。当一个 agent 需迁移到另一机器上时，它要调用一个"agent_jump"方法，该方法自动捕捉此 agent 的状态并将该状态发送给迁移目的地 server。此后，目标 server 将启动一个 Tcl 执行，载入该状态信息，从 agent 的中断点重新执行。

7 流动 agent 的标准化：MASIF

显然，上述流动 agent 系统在体系结构和系统实现上都存在较大差异，这种现状极大地妨碍了流动 agent 的互操作及技术的推广应用。因此，Crystaliz 公司、General Magic 公司、GMD 公司、IBM 公司以及 Open Group 等单位共同提出了流动 agent 的标准化草案：Mobile Agent System Interoperability Facility（MASIF），并引起了 OMG 的注意。

MASIF 定义了 agent 系统之间而非 agent 应用和 agent 系统之间的接口。对流动 agent 而言，语言互操作难度太大，故 MASIF 没有语言的互操作的内容，它只限于不同开发者用相同语言实现的流动 agent 系统间的互操作。此外，MASIF 也没有对局部的 agent 操作如解释、序列化、执行等进行标准化，因此，MASIF 实际上是一个在 agent 系统级上，而非 agent 本身级别上的接口标准化。MASIF 制定了以下四方面的标准：

agent 管理：从事流动 agent 的工作人员非常希望对 agent 管理进行标准化。很明显，一个管理着不同类型的流动 agent 系统的管理员肯定期望这些系统能在同一标准下运行。该标准下，我们可以创建一

（下转第 13 页）

(2)对任意记录 r,若 r 在 MC 缓存中,则 r 也必在 S_{CCR}中;

(3)对任意记录 r,若 r 在 S_{CCR}中,则 r 必在 MC 的缓存中。

我们采用了归纳法证明。定理 1 得证。

如果我们不要求服务器将已提交事务的删除操作记录为单记录删除的话,CS 缓存同步算法的正确性将不能保证。例如,MC 的缓存包括关系 ORDER。如果服务器上时间戳大于 MC_NTS 的已提交事务日志中包含一个更新操作:

```
update ORDER
    set QUANTITY=QUANTITY+5
```

随后又有一个集合删除操作:

```
delete from ORDER
    where QUANTITY≥50
```

则在完成 CS 缓存同步算法之后,MC 的缓存将出现漏删现象:由于 CS 算法先执行删除操作,因此在同步之前 MC 的缓存中 QUANTITY 在 45～50 之间的 ORDER 记录将不会被删除,而这些记录在服务器上都将被删除。因此,在记录已提交事务日志时,服务器必须将所有的集合删除操作转化为单记录删除操作序列。

小结 针对允许只缓存服务器数据库部分子集的移动客户机缓存机制,本文介绍了一种缓存同步算法—CS 算法。CS 算法以更新事务操作日志的传送为基础,保证了 MC 的缓存在经过同步之后仍然保持与服务器上数据库之间的一致性,并且继续维护缓存子集描述 CCD 的有效性。同时,CS 算法只要求服务器向 CM 发送自上次缓存同步以来服务器更新事务的删除操作日志以及被更新的数据库记录,因而尽可能地降低了服务器与 MC 的处理开销,具有很高的执行效率。因此,CS 算法是一个正确、高效的缓存同步算法。

参 考 文 献

1 李霖,周兴铭 移动数据库中客户机的断接操作.计算机科学,1999,26(3)

2 Demers A, et al. The Bayou architecture: support for data sharing among mobile users. In Proc. IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, California, 1994. 2～7

(上接第 4 页)

个 agent,给定一个名字,然后在一个标准的模式下挂起、恢复或终止这个 agent 的执行。

agent 转移:agent 应用中创建出的众多 agent 在不同类型的 agent 系统中自由移动的能力是标准化的主要目标之一,它规定了一个公共的基础构架。

agent 和 agent 系统的命名:agent 系统的互操作除了要求 agent 操作的标准化,其参数的语法和语义也必须标准化,特别是 agent 和 agent 系统命名也要标准化,它使得应用能够标识 agent 和 agent 系统,亦使得 agent 和 agent 系统能互相标识。

agent 系统类型和定位:只有能被某一目标 agent系统所支持,agent 才能向该 agent 系统迁移,因此,agent 迁移前必须要能从异地获取目标系统的类型信息,故 agent 系统的定位语法必须标准化。从 agent 系统彼此定位的角度来讲,定位语法也需标准化。

小结 本文的中心议题是 agent 的流动特性,我们认为流动 agent 是一独立的计算机程序,它可自主地在异构的网络上,按照一定的规程流动,寻求合适的计算资源、信息资源或软件资源,利用与这些资源同处一台主机或网络的优势,处理或使用这些资源,代表用户完成特定的任务。此外,我们列出了使用流动 agent 所带来的若干好处,给出了 agent 技术的几个前景较好的应用领域,并介绍了若干目前较为成熟的基于 Java 的流动 agent 系统。尽管基于 Java 的流动 agent 系统有许多共性,但不能实现互操作,为此本文最后介绍了 OMG 提出的 MASIF:流动 agent 技术的某些成份的标准化。agent 技术的标准化对该技术的影响将会是令人鼓舞的。(参考文献,略)