MAS环境下实现Agent交互协作的关键技术

王 珑* 章 毅 吴 跃

(电子科技大学计算机学院 成都 610054)

【摘要】讨论了MAS环境下实现Agent交互协作的关键性问题。在分析Agent协同设计目标的基础上，将遗传算法应用于子任务的调度，提高了Agent协同工作的效率。应用Agent的通信语言建立了一个基于知识查询与操纵语言的Agent通信模型，通过引入Agent位置追踪和Agent消息缓存机制解决了Agent移动过程中遇到的位置追踪和通信失效问题。

关 键 词 多主体系统; 移动代理; 知识查询与操纵语言; 通信模型; 协作

中图分类号 TP31 文献标识码 A

Key Techniques to Realize Cooperation of Mobile Agent in MAS Environment

Wang Long Zhang Yi Wu Yue

(School of Computer Science and Engineering, UEST of China Chengdu 610054)

Abstract In MAS environment, some key techniques are needed to realize interaction and cooperation of mobile Agents. In this paper, genetic algorithm is introduced to assign Agent to a suitable node, improved the coordination efficiency of Agents. An Agent communication language—knowledge query and manipulation language is analyzed and an Agent communication model is established based on knowledge query and manipulation language. Also, an effective way to solve Agent location tracing and invalidated communicating during Agent roaming among network nodes is provided.

Key words multi agent system; mobile agent; knowledge query and manipulation language; communication model; cooperation

移动Agent技术是一种新兴的技术，它可以有效地简化分布式系统的设计、实现和维护。采用移动Agent计算模式可以有效地降低分布计算中的网络负载，提高通信效率[1~3]。MAS(Multi-Agent System)是由自主的Agent通过协作完成某些任务或达到某些目标的计算系统，它通过竞争或磋商等手段协调解决各Agent成员的目标和行为之间的矛盾与冲突。在MAS中，协作不仅能提高单个Agent以及由多Agent所形成的系统的整体行为性能，增强Agent及Agent系统解决问题的能力，使系统具有更好的灵活性[3,4]。资源、能力的有限性和分布性使协作成为MAS的重要特征之一。

Agent间进行协作的基础是交互，而通信是实现Agent交互的一种重要方式和手段。采用何种通信模式和语言关系到能否有效地实现Agent的交互协作。把Agent应用于网络环境中，还需要解决由Agent的移动性所带来的问题，如：如何将代表用户任务的Agent分配到适宜的网络节点中执行，以得到较高的效率；如何实现Agent的位置透明通信[5,6]等。

1 子任务调度算法

多Agent协同设计的基本目的是使参与设计的Agent各尽所能，充分利用网络中的分布资源，故要将协同

2002年9月10日收稿

* 女 26岁 硕士生 主要从事多主体系统理论及应用、数据库技术及应用方面的研究

任务进行分解，将分解后的子任务分配给合适的Agent求解，这成为实现协同设计的一个重要问题，其关键技术是选择何种任务调度算法。

在网络中，任务与资源的组合随子任务的增加呈指数增长，在实际的协调设计中，还需兼顾到实时性，因此不能盲目搜索算法。当搜索空间过大时，不宜使用启发式搜索方法。处理问题具有柔性和并行处理能力的遗传算法可以在较大的状态空间中随机高效地采样、搜索，很快逼近最优解，故该算法适合作为子任务分配的调度算法[7]。

遗传算法是一种求解问题的高效并行全局搜索方法，与其他搜索算法相比，它主要有以下特点：

1) 在搜索过程中能自动获取和积累有关搜索空间的知识，并自适应控制搜索过程以逼近最优解；

2) 搜索过程和优化计算不依赖于信息梯度，适合处理传统搜索方法难以解决的复杂非线性问题；

3) 可从任一初始化群体出发，通过各种遗传操作，逼近最优解。

遗传算法具有“生成+检测”的迭代过程，其基本处理流程如图1所示。

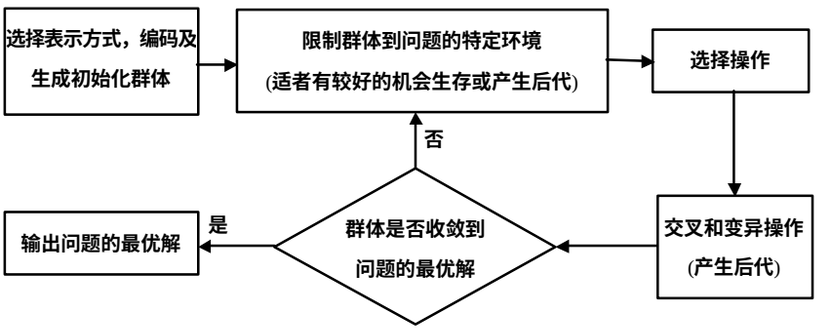

图1　遗传算法的基本处理流程图

遗传算法是一种群体型操作，它以群体中的所有个体作为对象，通过循环执行选择(selection)、交叉(crossover)和变异(mutation)三种操作，直到群体收敛到问题的最优解。

在遗传算法中，可采用多种形式来描述染色体，如二进制数字串、矩阵等。下面用一个二维矩阵$\boldsymbol{T}_N$来描述：

$$\boldsymbol{T}_N=\begin{array}{c} \begin{matrix} t_1 & t_2 & \cdots & t_m \end{matrix} \\ \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix} \end{array} \begin{matrix} node_1 \\ node_2 \\ \vdots \\ node_n \end{matrix}$$

矩阵中的元素a_{ij}称为基因，a_{ij}=1表示第j个子任务调度到第i个节点上。假设子任务个数大于网络节点数，故每个节点最少分配一个子任务且只能分配到一个节点上。所以$\boldsymbol{T}_N$矩阵应满足约束条件：

$$\sum_{j=1}^{m} a_{ij} \geq 1 \qquad i=1, 2, \cdots, n$$

$$\sum_{i=1}^{n} a_{ij} = 1 \qquad j=1, 2, \cdots, m$$

在任务调度问题中，先按约束条件生成一定数量的染色体，从中选取最好的个体加入初始群体，不断重复这一过程，直到初始群体的个数达到预定的数量。遗传算法以该初始群体为起点，通过对染色体中的基因进行各种遗传操作，逐代进化，直到获得一个满足要求的染色体即是子任务调度的结果。

当子任务个数较少，每个子任务可以单独占用一个网络节点时，称为粗粒度任务调度。粗粒度任务调度只需找到一个资源配置满足或接近资源需求的节点。当子任务个数较多，每个节点要处理多个子任务时，称为细粒度任务调度。在调度模型中，以子任务的个数相对于系统可调度节点数来衡量子任务的粒度。当子任务个数小于系统可调度节点数时，为粗粒度调度，否则为细粒度调度。粗粒度调度多采用Agent竞争法或预先指定法，而细粒度调度则采用遗传算法。

2 Agent通信语言

Agent通常应用于网络环境中，有时要跨多个平台，而网络中各个平台使用的语言不相同。为实现Agent间的通信，必须建立统一的Agent通信语言(Agent Communication Language，ACL) 。

作为Agent的通信语言，ACL须具备三个特征：

1) 非耦合性：要求在空间和时间上以非耦合方式协调Agent之间的交互。即当一个目标Agent并不存在时，另一个Agent也能执行发送；

2) 联合声明：不需要直接指明建立通信的Agent名称，而是通过向网络声明一个需求的模板，由对方根据相应的匹配机制找到符合要求的Agent；

3) 关系分割：要求由主程序语言导出的ACL不受主程序语言的影响。

Agent在应用ACL实现通信时应遵循三个准则：

1) 真实性：Agent应说实话；

2) 自治性：Agent应有一定程度的独立性；

3) 承担义务：如果命令某个Agent执行一个任务，它应当尽力完成。

知识查询与操纵语言(Knowledge Query and Manipulation Language，KQML)作为一种获得广泛应用的ACL，正在成为Agent之间通信的事实标准。KQML是一种交换知识和信息的描述性语言，它定义了Agent间传递消息的格式和消息处理的协议，通过提供一套标准的通信原语实现Agent间信息的交流和知识的共享。它既是一种Agent间消息的表示格式，也是一种处理消息的协议。

KQML是一种层次结构型语言，它可分为三个层次:

1) 内容层：消息所包含的内容，用相应的程序语言表示。该语言可用ASCII字符或二进制的形式表示。KQML语言的具体实现并不关心消息内容的具体含义；

2) 通信层：描述与通信双方相关的一组属性参数，包括通信底层参数，消息接收者和发送者的标识以及与通信相连的标识；

3) 消息层：为KQML的核心，主要确定消息传送所使用的协议，由发送方提供一个与内涵相关的原语，指明消息中的内涵为确认、询问、命令或其他的原语操作。

KQML的语法简单，为基于平衡的括号表。表开始为行为原语名称，其余部分为一组以“:关键字”形式出现的参数表，为实际应用中根据需要进行修改提供了方便。KQML定义了一组预留的含义明确的行为原语，开发人员还可以根据实际应用的需要添加满足KQML标准的新原语。

习惯上，一条KQML行为原语也称为一条消息。典型的KQML消息如下:

```
{
  ask-one
        :sender Tracy
        :content(PRICE fashion)
        :receiver Infoserver
        :reply-with location-shop
        :language self-defined
        :ontology fashion-mode
}
```

其中，KQML消息的行为原语是ask-one；

“：sender”表示的是消息的发送者；

“：receiver”表示消息的接收者；

“：content”表示消息内容；

“：reply-with”表示下条消息对本条消息的响应；

“：language”表示消息内容使用语言的名称；

“：ontology”表示消息内容使用的实体集的名称；

“：content”为内容层；

“：sender，：receiver，：reply-with”构成了通信层；

行为原语“ask-one，：language，：ontology”为消息层。

3　Agent通信模式与模型

通信模式与多Agent协作的组织方式紧密相关，通信模式的优劣关系到整个系统的效能。在实践中通常采用下面两种方法：

1) 直接通信法：由Agent自己负责与其他Agent的谈判。Agent实体中需集成进一定数量的专门处理谈判的代码，这使Agent实体的设计复杂化。它的另一个缺陷是当网络中Agent数量较多时，Agent之间建立通信连接的时间较长。

2) 辅助协作法：首先把Agent分类，将处于同一地域或一定社会中的Agent组成一个Agent群，在每个Agent群中设立一个专门用于通信的装置，此装置也由一组具有某种特定功能的Agent组成。Agent群中的Agent只与此装置通信，此装置根据请求的模板，查找匹配的通信对象。当同一个Agent群中存在适合的匹配对象时,可以直接通过此装置建立通信。但当匹配对象在其他Agent群中时,通信过程就必须加入与其他Agent群中的装置之间的通信。多种通信方式的结合形成了一个Agent群的联盟，它不但解决了Agent自己管理通信时复杂的设计，也解决了当Agent数量太大时直接通信耗时多的问题。

下面是一个基于辅助协作法的多Agent通信模型。Mobile Agent需要在一定的环境中运行，这种环境称为码头(dock)。它不但使Agent可驻留于网络中某节点上，访问该节点的资源，还提供了Agent与其他Agent进行通信的场所。

Agent的交互行为如图2所示。若一个对象有自己可控制的线程称为主动对象，否则称为被动对象。Agent将一个内嵌在KQML中的消息，通过码头提供的服务，发送给接收对象(本地对象或其他Agent)。在图2的左边部分，Agent是主动的，在KQML的发送和接收过程中，Agent还可与其他Agent或本地对象进行通信。码头和本地对象是由Agent的行为引起的一种被动反应，是被动对象。在图2的右边部分，两个Agent都是主动的，在它们通信的过程中，均可与其他的Agent进行通信。Agent的典型生存期有三个阶段：

1) Agent在某个码头上被创建；

2) Agent访问一个或多个码头；

3) Agent终止。

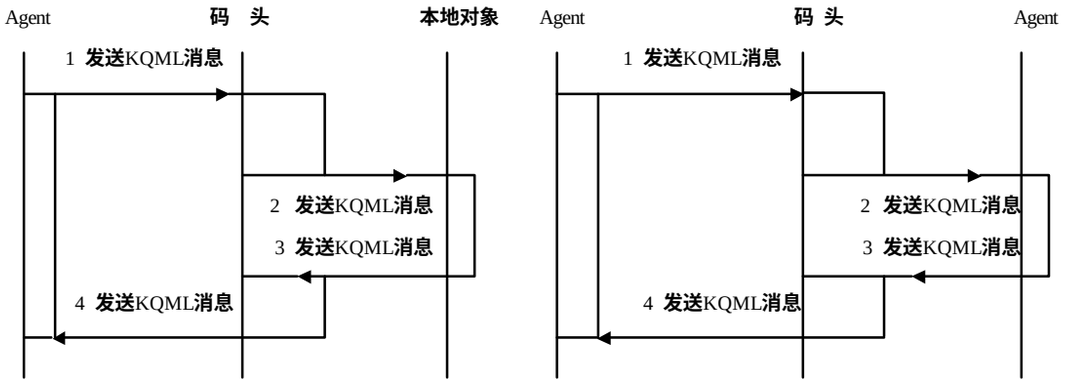

图2　Agent的交互行为

在Agent访问一个码头的过程中，它利用KQML消息与码头、本地对象或其他Agent进行通信，码头则提供给Agent各种服务，如路由选择、安全检查、Agent迁移等。Agent最常见的行为是与本地对象或其他的本地码头上的Agent进行通信。

Agent的通信模型如图3所示。图中的通信模型主要有Agent、码头和本地对象三种。本地对象是静态的能够理解KQML消息的对象，码头是提供给Agent和本地对象的服务场所，每一个Agent只可同时访问一个码

头。Agent通过发送和接收KQML消息这种唯一的通信方式与其他对象进行通信。KQML 消息通常由码头进行路由选择，使码头对消息进行安全的授权检查。码头不必知道或解释KQML消息的内容，使它对具体的应用细节透明，防止了可能对Agent的改变，提高了系统的安全性。

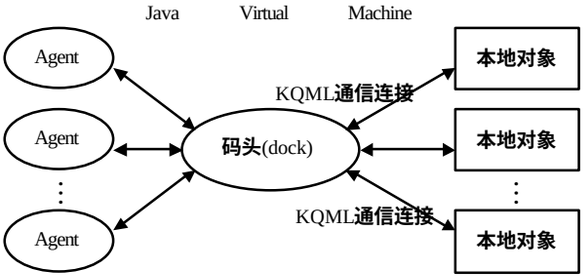

图3 Agent通信模型

Agent和本地对象同时只能与一个码头相连接。如果一个Agent希望同与另一个码头相连的本地对象通信时，它必须离开当前的码头迁移到另一个码头上。即KQML通信只能在同一个JVM中进行，所以，Agent要与其他的JVM 中的对象通信，必须进行迁移。

4 实现Agent位置透明通信

在基于MAS的网络环境中，要实现Agent间的通信，接收消息的Agent的地址必须明确。而Agent的位置在网络中是不断变化的，使对Agent的寻址较困难。但无论采用由发生位置迁移的Agent主动通知其他Agent的方式或其他Agent通信前进行查询的方式来解决对Agent的寻址，都会增加网络和Agent的负担，较理想的解决方法是在MAS环境中设置一个专门提供Agent位置服务的机制 ——Agent 位置追踪(Agent Location Tracing，ALT)。

目前所采用的Agent位置追踪方法主要有三种：

1) 树状域追踪：将MAS环境中的每一计算机系统看作一个叶子节点域，由它提供名字服务，建立当前域内所有Agent与其所在位置的映射关系表。当Agent被创建或发生位置改变时，通过修改相应域的映射表实现对Agent的位置追踪。

2) 分布式追踪：不提供名字服务，利用“前指链”来实现Agent的位置透明。即Agent每发生一次位置变化，都会在前一个驻留点留下一个指向当前位置的指针，发送者只要找到该Agent的创建位置即可循着该前向指针链找到目标Agent。

3) 由Agent创建地提供地址：当Agent发生位置改变时，系统在该Agent创建地记录其当前位置，消息均发送到创建地，若该Agent不在创建地，则由创建地转发。

采用任一方法，消息发送Agent通过ALT就可找到消息接收者的当前位置，完成消息的传送或转发，实现了对Agent透明寻址。

在移动Agent的通信过程中，经常会发生通信失效的情况。如：Agent 1 要向主机*A*上的Agent 2发送消息，但在消息发送过程中，Agent 2从主机*A*迁移到了主机*B*，该消息因无法正确找到接收者，而可能丢失，此现象称为通信失效。通信失效是Multi-Agent系统中Agent协作的致命缺陷，它使协作中的Agent不能及时或无法正确接收协作消息，导致协作失败。对于这个问题，如果规定在Agent迁移过程中拒绝接收任何消息，容易增加发送方的负荷并引起消息丢失。较好的解决方法是在MAS系统中增设一缓存机制(Agent Message Cache，AMC)，在Agent完成迁移后，再将消息转发给它。具体实现过程如下：

1) Agent迁移到某位置时，由该位置为其分配一消息代理，负责处理发送给Agent的消息，代理的生命周期和Agent在该位置的生命周期相同；

2) 在Agent的迁移过程中，代理的生命周期并未结束，它仍然负责接收发送给该Agent的消息，并缓存起来；

3) Agent迁移成功后，由新位置为其分配消息代理，前任代理将缓存的消息打包转发给新代理，然后结

束自己的生命周期；

4) 若Agent迁移失败，仍由原代理负责接收消息。

因此在Agent的整个生命周期中，始终有且仅有一个代理负责接收发送给该Agent的消息，因而不会出现通信失效的情况。

5 结 束 语

为了促进MAS的开发、推动Agent间的协作计算以及相关问题的解决，本文就移动Agent在MAS环境下实现交互协作的几个关键问题做了一些探讨。目前，多Agent交互协作技术还不成熟，有许多问题有待深入研究，包括系统的结构和管理、设计任务的规划和分解、设计知识的表示和转换、主体的通信策略和语言、冲突的识别和消解、系统的安全性等。随着对多Agent系统理论与应用研究的不断深入，多 Agent技术也将日益得到完善，它将成为未来网络环境的主流技术。

参 考 文 献

[1] Wooldridge M J, Jennings N R. Intelligent Agent: theory and practice[J]. Knowledge Engineering Review 1995, 10(2): 115-152

[2] 马俊涛, 刘积仁. Mobile Agent体系结构及关键技术探讨[J]. 小型微型计算机系统, 1998, 19(2): 7-14

[3] 史忠植. 智能主体及其应用[M]. 北京: 科学出版社, 2000

[4] 李　强, 吴泉源, 王怀民. 一种Agent互操作语言的设计[J]. 计算机学报, 1998, 21(8): 213-217

[5] 陶先平, 吕　建, 张冠群, 等. 一种移动Agent结构化迁移机制的设计和实现[J]. 软件学报, 2000, 11(7): 918-923

[6] 郭瑞景, 苏　敏, 陶先平. 基于KQML的Agent安全通讯模型[J].小型微型计算机系统, 2001, 22(10): 1 192-1 195

[7] 席裕庚, 柴天佑, 恽为民. 遗传算法综述[J]. 控制理论与应用, 1996, 13(6): 697-708

编　辑　漆　蓉

上接第148页

4 Conclusions

In this paper, we have presented a new design of low-noise low-power consumption charge amplifier. Simulated by EDA software Cadence, the results obtained are satisfied. The DC open-loop gain is 82.9 dB with a 28 kHz –3 dB bandwidth and its phase margin is 46.9°. And the maximum output noise spectral density is 1.5 $\mu V/Hz^2$ at very low frequency. Using standard 3 μm P-Well CMOS technology, the proposed amplifier is fabricated, and the measurement results are closed to the simulation. Due to its good performance, this kind of charge amplifier can be widely used in particle physics, nuclear physics, and γ-ray detection.

Reference

[1] Radeka V, Recia S, Manfredi PF. JFET monolithic preamplifier with outstanding noise[J]. IEEE Trans On Nuclear Science, 1993, 40(4): 744-749

[2] Christian M, Huang Qiuting. A low-noise CMOS instrumentation amplifier for thermoelectric infrared detectors[J]. IEEE JSSC , 1997, 32(7): 968-976

[3] Hu Y, Solere J L, Lachatre D. Design and performance of a low-noise, low-power consumption CMOS charge amplifier for capacitive detectors[J]. IEEE Trans On Nuclear Science, 1998, 45(1): 119-123

[4] Binkley D M, Puchett B S. A power-efficient, low-noise, wideband, integrated CMOS preamplifier for LSO/APD PET systems[J]. IEEE Trans On Nuclear Science, 2000, 47(3): 810-817

[5] Chang Z, Sansen W M. Low-noise, low-distortion CMOS AM wide-band amplifiers matching a capacitive source[J]. IEEE JSSC, 1990, 25(6): 833-840

编　辑　漆　蓉