计算机科学2002Vol. 29№. 1

信息 Agent 的体系结构与行为设计

Architecture and Behavior Design for Information Agents

郑淑丽　杨敬安　骆祥峰

(合肥工业大学人工智能所　合肥230009)

Abstract With the rapid development of Internet, information Agent aroused a great interest with its potential power. Various information agents were made for different targets and were implemented in different means. To facilitate open system interoperability of autonomous agents and reduce program load of programmers, we need to specify reusable architecture to support some reusable behavior for information agent. First, the function overview and the basic architecture of information agent are stated, and then the most important modules are discussed. They are the schedule system and the local DBMS. Finally, five reusable behaviors are presented.

Keywords Information agent, Local DBMS, Reusable behavior

1　引言

Internet 的迅速普及和 WWW 技术的发展，把人类迅速推向信息化社会。面对庞大的、瞬息万变的信息资源，Internet 上现有的搜索引擎(如 Yahoo，AltaVista)和邮件管理软件已经不能满足用户的需求。"信息过载"和"资源迷向"已成为 Internet 信息服务质量的瓶颈[1]。在这种情况下，信息 Agent 技术被提了出来，它集软件、通信、分布系统的技术于一体，进行网上信息资源管理、控制和分类，如资源导航、用户问题解惑、信息过滤和筛选以及知识发现，从而改善信息服务的质量和水平。

信息 Agent 系统的研究吸引了众多研究机构和工业组织的注意力。目前，较为成功的信息 Agent 的应用范例有网上信息检索、过滤系统(如 Musag 系统[2]，Amalthaea[3]系统)，网上广告业务以及邮件服务(如 Magi[4]系统)等。随着 Web 上信息 Agent 数目的快速增长，我们需要制定出一个合理的 Agent 结构框架，使得：(a)无须采用特别的方式来重新构建每个 Agent；(b)能够促进由不同设计者开发的信息 Agent 之间的交互和协作[5]。

为达到上述目的，我们可以规定信息 Agent 的基本体系结构及行为，这样不仅极大地减轻了程序员的编程负担，而且有利于信息 Agent 之间的交互。

2　信息 Agent 的体系结构

信息 Agent 作为信息处理的智能体，为任务 Agent 提供信息服务[6]。信息 Agent 与 HTTP 有着明显的区别。它能够对任务 Agent 的请求进行规划调度，动态建立本地信息，即时获取外部信息，并有序地完成请求任务。

2.1　信息 Agent 的基本功能

信息 Agent 的主要功能包括：处理一般查询请求(Common Request)；提供定期的信息服务(Repeater)；根据给定的信息模式来监控外部信息资源(Reminder)。总而言之，信息 Agent 根据即时和预先存储的请求访问本地或外部信息资源。

2.2　信息 Agent 的组成部分

如图1所示，信息 Agent 主要由三个组成部分：任务请求处理单元(任务请求集和任务分解调度器)、本地信息库以及外部接口。任务请求处理单元负责接收请求信息，分析信息服务种类、要求响应时间，调度任务的执行以完成信息请求并监控当前操作。Agent 根据任务请求处理单元调用外部接口程序和调节本地信息库的内容。

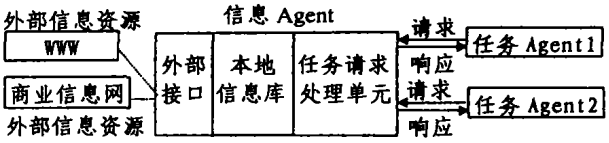

图1　信息 Agent 的功能示意图

本地信息库提供所有用于响应任务 Agent 请求的信息。与传统的信息库不同，本地信息库可以根据请求产生动态的数据定义语言(DDL)进行自我维护；根据任务分解调度单元的指示以决定数据的存储期限和组织方式。本地信息库是与具体应用无关的信息库。

外部接口为本地信息库提供长期和临时的数据。外部接口接收来自任务请求处理单元的任务分解调度器的处理请求，从外部信息资源获取数据，并将结果返回本地信息库。外部接口是不可重用的。

2.3　信息 Agent 的任务调度系统

图2给出了信息 Agent 内部的数据流和控制流走向。信息 Agent 的任务调度系统主要解决如下问题：1)任务 Agent 需要什么样的信息?2)如何获取所需信息?3)本地信息库当前有什么信息?4)需要从外界信息库获取什么样的信息?

为了完成上述功能，首先要有一个任务请求集，任务请求集是被解释过的信息检索任务，主要反映任务的种类、需求和时限等信息。任务请求集除了包含即时请求信息，也包含定期请求和监控信息(Repeater and Reminder)。任务请求集是任务分解调度器的数据源。

任务分解器根据任务请求的优先集，将任务压入当前处理队列。任务分解器根据任务分解库，逐层获取分解和归并相关任务(相关任务即在任务分解过程中产生的依赖性任务)并最终获得独立和相关任务集、执行路径和执行代价。

任务调度器根据任务上述结果决定数据的获取和组织方法。可以描述为：

1)由于信息的可重用性，本地数据库尽可能将数据保存

在本地数据库，这就是说，独立或相关信息可能在本地数据库存在。所以，任务调度器首先获知是从本地信息库获取信息还是调用外部接口程序。

2)调度器根据任务队列的执行总代价决定使用前推或后压的方式决定执行时间表，并随时根据新的任务，考虑现有的信息资源和处理能力，修改执行时间表。调度器总是保证每个任务在其要求期限内完成，但将放弃因网络拥塞等原因超出执行期限的任务。

3)由于本地数据库存储空间的限制，调度器将在达到临界存储值时删除存储权最小查询结果集。并且，调度器保证任务集合中的请求在被响应之前不被删除。

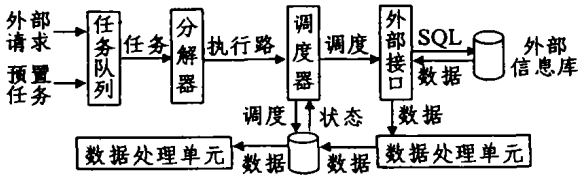

图2 信息 Agent 内的数据及控制

2.4 信息 Agent 的本地信息系统

由于本地信息库存储的信息与特定领域无关，即数据结构不能预先面向应用设计。本地信息库是一个面向对象的一元数据库，主要支持以下对象类型：

1)简单查询(Single Query)对象，该对象的基本属性是编码、查询原语、存储权值和时间戳信息。简单查询对象可以被继承。

2)行对象，该对象的基本属性是行编码、行号和查询编码。

3)列对象，该对象的基本属性为列编码、列名、列数据类型(包括：数值，字符串，时间和大型文本对象)、列长度和行编码。

4)信息元对象，该对象的基本属性为信息编码、值(表现为变长字符)和列编码。

上一节提到，任务分解器将标准查询分解为查询原语，查询原语将获得相关的数据元信息并动态地通过查询自身确定的数据定义语言(Data Definition Language，DDL)动态组成中间结果集，该结果集在一个查询事务内保持，并在未观测到数据元变化之前可以被重用。

查询原语是信息 Agent 中最重要的设计，它能够实现 ANSI-92标准中的所有数据操纵语言，这些原语体现了简单查询的基本元素(在这里关联、子查询等操作被任务分解器分解，而最终由任务调度器合并)，例如：

Select Column(选择列)
Full Scan(扫描全部数据元)
Index Range Scan(根据数据元值索引扫描数据元)
And Equal(行编码合并)
Filter(筛选行编码)
Count(统计行编码)

本地数据库的设计虽然在原则上与领域无关，但任务分解器必须了解本地数据资源和外部数据资源的数据定义，从这个意义上说，任务分解器的处理知识库必须要面向特定数据结构。而本地数据库和任务分解逻辑都是可以重用的部分。

3 信息 Agent 的行为设计

所谓行为，指的是信息 Agent 完成目标的方法。我们可以用一个具体的任务实例、子任务集合(或简单的操作)以及子任务之间的信息流关系来表示某种特定的行为。对于一个具体的任务，信息 Agent 的规划程序根据本地信息库来分解该任务并决定子任务之间的信息流关系，然后将子任务提交给信息 Agent 的调度程序。调度程序根据信息流关系来决定控制流。这里主要讨论信息 Agent 的五种可重用行为，分别是：定义行为，信息查询，信息监控，自我调整以及自我复制。

3.1 定义行为

信息 Agent 在刚建立的时候，需要向外部数据源获取它们的数据结构，以便将其提供给任务 Agent 并修改任务分解和调度的知识库。

定义行为包括两个基本操作：①获取外部信息源并获取其信息的数据结构定义；②根据数据结构定义修改知识库。

信息 Agent 向外部数据源发送信息共享请求，要求提供数据结构定义。在对方获准后进行数据结构定义获取。信息 Agent 在获取定义的同时，也获得外部数据源的响应时间、存储成本和数据稳定性，以作为该信息的初始权值。如果数据的稳定性较高(相对静态)而存储成本较小，信息 Agent 将直接复制数据，存储到本地信息库的数据元中。所有的信息 Agent 都可以共享(或称为重用)定义行为。

3.2 信息查询和信息监控

信息查询是最简单的信息 Agent 行为，信息查询分为即时查询和周期性查询。而信息监控略有区别，它在信息发生变化时发出提醒(因为信息 Agent 主动接受外部数据并且不面向应用设计，所以，对于 Agent 而言，它和同周期性查询一样属于预定义任务。不同的是，信息监控所要求的查询频率更高，而响应优先级较低)。

无论是信息查询还是信息监控，都存储于任务请求集等待响应。信息 Agent 对任务请求集中的检索要求进行分解，分析信息的各个获取步骤，哪些步骤的信息已经存储于本地信息库，哪些必须要从外部信息库获取。本地信息的获取通过信息 Agent 定义的原语实现，而外部信息的获取则通过调用外部接口，最终转换成数据元存储于本地信息库。信息 Agent 调度器监控各个步骤的完成情况，并把它们作为下一步骤的数据元。

信息查询执行成功后(即完成最后一个步骤后)，信息 Agent 将信息元按任务 Agent 所要求的格式重组数据元。

3.3 自我调整

信息 Agent 必须能够确定在数据库中应该存储什么样的数据元。信息 Agent 为本地数据库的存储情况设置一个临界点，当到达这个临界点而有新的存储请求时，信息 Agent 必须删除本地数据库中存储权值最小的数据，直到新的数据可以得到存储。

存储权值(Storage Weight Value)决定于信息使用的频率(Usage Frequency)、获取代价(Acquirement Cost)和存储代价(Storage Cost)，计算公式如下：

$$SW_v = f_v * C_a / C_s$$

存储权值面向查询结果集定义，是查询对象的一个属性。信息 Agent 维护着一个查询对象及其权值的列表，不断更新查询的权值并改变查询在列表中的位置。同样，信息 Agent 可以接受外界干预，将某个查询置于列表的顶部或底部。

3.4 自我复制[5]

信息 Agent 必须能够意识到所处的环境及自身状态，在过载的情况下能够采取相应的措施。为此，信息 Agent 可以

(下转第95页)

映像而不会引起名字冲突；

iii）如果不同的语言是在彼此的顶上被分层的，则应建立MPI界面的不同语言约束的实现文档以便形象的开发者知道形象界面是否必须对于每个约束来给出实现，或者是否仅须对最低层的程序来给出实现；

iv）确保不同语言约束的实现在哪里通过分层方法实现，则包装函数（例如Fortran的约束是一组包装函数，它们调用C的实现）要和这个库分开；

v）MPI库中提供一个不可操作的程序MPI-PCONTROL。

8. *对于Fortran和C的约束* MPI-2在这个基础上加上：①动态进程管理；②输入/输出；③ 一边的操作；④对C++的约束。

五、我国的对策

人类社会生活的各个方面，包括科学活动在内，都需要有秩序地进行。各种法规，规范，标准也就为此而制定的。然而，在很多情况下，这类东西又是强者强加于弱者的。在计算机科学技术领域内，我们大体上已经经历了四次标准的制定：(1)各种程序设计语言的标准。我们由于种种原因，无缘参与这项工作；(2)操作系统标准的制定，这主要是由国外大计算机企业或研究单位完成的，我们也无资格参加；(3)通讯协议的制定。在这方面，我们或许参加了一点工作，但由于我们的客观条件的限制，我们也没有多少发言权；(4)软件工程的规范。这次，MPI标准虽然涉及的仅是并行计算或分布式计算的问题，但由于在未来发展中，这是一个极为重要的领域，我们不能再置身事外，必须认真研究我们的对策。据我个人看来，对策不外有三：

一是完全不理会国外的什么标准，我们按自己的需要自搞一套。然而，这条路走不通，我们不可能不同国外发生联系，一旦我们的并行机要和国外的并行机实现通讯，就不可避免地要用已有的MPI标准。就是我们所购买的国外的并行机（这类机器的数量在我国肯定将不断增加），也还是要使用已有的MPI标准。

二是完全用国外的标准。这条路最省事，其实我们早也就司空见惯了这样一种情况。我们这么多年来，不就是什么都用人家的标准吗？然而，随着我国的并行机的发展以及在它上面应用的扩大，而且当我们的通讯是用中文信息进行时，就势必会发现，国外的标准未必完全符合我们的要求。因此，这就决定了完全采用国外的标准不可取。

三是原则上采用国外的标准，或者说参考国外的标准，但要根据我们中文信息处理以及将来中文计算的需要，进行适当的修改。这里有以下的问题：MPI-1和MPI-2所包括的这些内容，哪些完全符合我们的需要？凡属符合我们的需要的，我们自然无需作任何改动；而哪些，对于我们的中文信息通讯或计算，是不大适当的？对此自然应作适当修改；还有哪些根本不适合我们的需要？我们应当以我们真正需要的东西来代替它（不换也可，但就是不使用它）。

事实上，MPI讨论会在这个问题上，也采取了明智的态度，即有许多特征，他们考虑暂不列入标准中。这是因为感觉到对于这些特征还缺乏足够的经验，没有明确的判断；有些特征在他们看来，其应用有限；还有一点，就是担心增加了它们，使程序员实现它们的负担增加了，但是对于用户价值又不很大，当我们来考虑自己的标准时，对于这些特征也都要充分考察。

总之，MPI标准的制定，为我们开拓我国的并行计算和分布式计算，将打下良好的基础。这也是我们参与到世界性的标准制定的一次大好机会，我们应该站在面向未来的高度，搞出一个与国际接轨又符合我国需要的MPI标准。

参 考 文 献

1 苏运霖. 分布式系统与分布式算法. 暨南大学出版社，1995
2 Tel G. Distributed Algorithms. 剑桥大学出版社，1994
3 Sair M, et al. MPI—The Complete Reference, MIT Press, 1998. 1, 2
4 Gropp W, et al. Using MPI, Portable Parallel Programming with the Message-Passing Interface. MIT Press, 1994

（上接第43页）

建立一个简单的模型，该模型描述了在规定期限内完成目标与当前查询及任务特征是如何相关联的。然后，把该模型与新增一个信息Agent的假想局势做比较，如果评估结果认为有必要新增一个信息Agent，则该Agent暂停信息服务（向中介发送停止提供服务的信息），进行自我复制。

总结和展望 信息Agent是多Agent系统的重要组成部分，本文主要介绍了信息Agent的体系结构和五种可重用行为。合理的体系结构能够支持许多复杂的自治Agent行为。行为重用意味着信息Agent的行为与特定的领域知识无关。我们可以不需要编程或只须写几行代码就可以迅速创建一个新的信息Agent，甚至，当我们对一个信息Agent进行性能改善或增加新行为时，所有其它领域的信息Agent都可以从中获益。本文讨论了五种可重用行为：定义行为、信息查询、信息监控、自我调整以及自我复制。

对于信息Agent，仍有很多问题需要进一步研究，如多媒体信息检索与服务，分布式运行的信息Agent合作问题的研究[7]，以及信息Agent学习算法[8]等。信息Agent相对于传统的搜索引擎而言，其性能有了很大的改善。信息Agent在没有用户的指导的情况下，能够对外部请求进行思考并有序地完成请求任务。为使信息Agent更加适应网上应用，为用户提供更加高质量的服务，有必要对信息Agent进行更深入的研究。

参 考 文 献

1 Takahashi K. Intelligent pages: collecting shop and service information with software agents. Applied Artificial Intelligence, 1997 (11): 489～499
2 Goldman C V, et al. Musag: An agent that learns what you mean. 1997
3 Moukas A. Amalthaea: Information discovery and filtering using a multi-agent evolving ecosystem. Applied Artificial Intelligence, 1997(11): 437～457
4 Payne T R, Edwards P. Interface agents that learn: An investigation of learning issues in a mail agent interface. Applied Artificial Intelligence, 1997(11): 1～32
5 Decker K, Pannu A, et al. Designing behaviors for information agents. In the Robotics Institute, Carnegie-Mellon University 1996
6 Gudivada V N. Information retrieval on the World Wide Web. IEEE Internet Computing, 1997, (5): 58～68
7 Oates T, Prasad M V N, et al. A distributed problem solving approach to cooperative information gathering. In AAAI Spring Symposium on Information Gathering in Distributed Environments, Stanford University, March 1995
8 Decker K S, Lesser V R. Designing a family of coordination algorithms. In: Proc. of the First Intl. Conf. on Multi-Agent systems. AAAI Press, San Francisco, June 1995