第 36 卷 第 5 期 计 算 机 工 程 2010 年 3 月
Vol.36 No.5 Computer Engineering March 2010
·软件技术与数据库· 文章编号：1000—3428(2010)05—0067—03 文献标识码：A 中图分类号：TP311

多 Agent 协同处理模型的研究与设计

徐 慧[1,2]，杨永国[2]
(1. 中国矿业大学计算机科学与技术学院，徐州 221116；2. 中国矿业大学资源与地球科学学院，徐州 221116)

摘 要：为能快速有效地处理开放式网络环境中的任务，结合移动 Agent 技术和计算机支持的协同工作技术，提出多 Agent 协同处理模型，并对其实现的相关技术进行研究。由协作管理器对任务进行分解，对负责处理子任务的 Agent 进行控制，使 Agent 之间能够进行协作处理，应用有效的算法为分解的子任务建立其最优执行序列，提高任务的执行效率。

关键词：智能体；协同处理；子任务

Research and Design of Multi-Agent Cooperative Processing Model

XU Hui[1,2], YANG Yong-guo[2]
(1. School of Computer Science and Technology, China University of Mining & Technology, Xuzhou 221116;
2. School of Mineral Resource and Earth Science, China University of Mining & Technology, Xuzhou 221116)

【Abstract】In order to process the task in the open network environment quickly and efficiently, this paper presents the Multi-Agent cooperative processing model combining mobile Agent with computer supported cooperative work and researches on the implementing technologies in the model. Collaboration manager which can decompose task controls the mobile Agents to realize the cooperation between them. An effective algorithm is applied to model to get the best schedule of subtask to reduce the task processing time and improve efficiency.

【Key words】Agent; cooperative processing; sub-task

1 概述

目前大部分网络应用构建在开放的分布式环境中，大多基于 C/S 或 B/S 体系结构，采用分布式对象技术，如 CORBA, DCOM 或 Java/RMI。但是随着网络环境的日益复杂，它们无法适应开放式的网络环境和各种复杂、动态的分布计算要求。近年来，基于移动 Agent 的分布式网络环境下的应用得到了较快发展。移动 Agent 的跨平台移动、动态和分布计算等特性扩展了传统 Agent 处理事务的能力，使之能较好地解决 Internet 环境中低带宽和不稳定连接的问题，并可提供对网络中非持续性连接设备上网的支持。

计算机支持的协同工作(Computer Supported Cooperative Work, CSCW)需要建立多个平等实体的交互关系，强调系统中各实体的协调合作。Agent 间的通信行为和特性与协同工作中群体成员的交流行为相适应。Agent 之间是对等的实体，存在的是平等协作的关系，强调 Agent 间通过通信而协作完成任务。因此，由 Agent 自主性引发出来的协同性使得 Agent 可以更好地反应协作群体或组织的结构和联系特征。本文研究多 Agent 之间的协同工作，提出了多 Agent 协同处理模型。

2 多Agent协同研究简介

20 世纪 90 年代初，General Magic 公司在推出其商业系统 Telescript 时提出了移动 Agent 的概念，它是一个能在异构网络环境中自主地从一台主机迁移到另一台主机，并可与其他 Agent 或资源交互的软件实体[1]。要确保多 Agent 系统协同产生一致的行为，主要涉及以下几个方面：

(1)组织：通过对角色、行为期待、授权关系等定义，组织为 Agent 协同提供框架。一组 Agent、由 Agent 完成的活动、Agent 之间的连接、目标、可以评估的进化标准等组成一个组织。组织结构对 Agent 的通信与协同方式加以约束。

(2)任务分配：任务分配涉及赋予 Agent 责任和问题求解资源分配等。对任务相关性最小化可以减少协同中的通信负担和潜在的冲突，从而有效解决问题。分解后的子任务之间在逻辑上存在一定的关系，而这些关系最终体现在 Agent 的任务执行过程中。因此，Agent 之间的关系体现了各个子任务之间的联系，两者存在映射关系。

(3)多 Agent 规划：Agent 通过规划其行动来改善协同中的一致性。在多 Agent 系统中要考虑其他 Agent 活动对 Agent 选择行动的影响、Agent 对其他 Agent 承诺对自身行为的影响以及其他 Agent 引起的难以预测的环境变化。

(4)通信管理：通过规划内容、数量、类型、通信时间，Agent 可以实现协同和问题的解决。

(5)协同关系：在协同环境中，Agent*i* 与 Agent*j* 的协作关系称为一个原子协同关系，简称为原子关系。基于任务分解的 Agent 之间的原子关系是具体和唯一的。

3 多Agent协同处理模型

由于可以通过强调 Agent 间通过通信进行协作来完成任务，因此对于分布在网络中的多个 Agent 来说，进行有效的协作是一个重要问题。本文提出一种多 Agent 协同处理模型(Multi-Agent Cooperative Processing Model, MACPM)，如图 1 所示。MACPM 体系结构共分 4 层：数据层，服务层，管理层和应用层。其中，应用层主要提供用户接口，方便用

作者简介：徐 慧(1980－)，女，讲师、博士研究生，主研方向：协同处理，虚拟地理环境；杨永国，教授、博士生导师
收稿日期：2009-07-30 **E-mail：**xuhui@cumt.edu.cn

户提交任务。

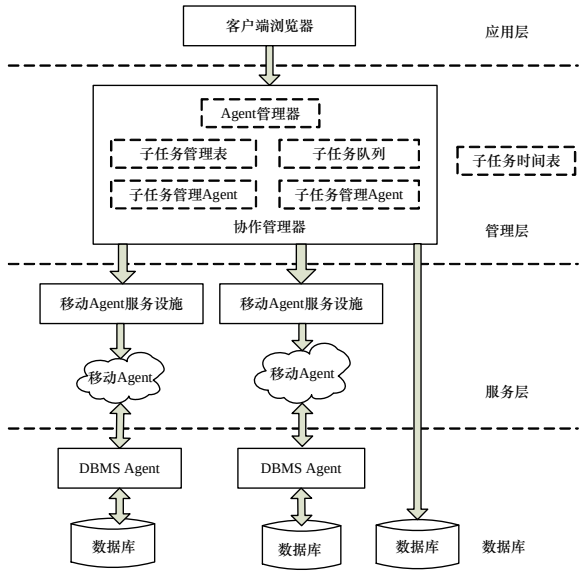

图 1　MACPM 模型

3.1　数据层

数据层包括：

(1)数据库：存放任务处理过程中的各种相关数据。

(2)元数据库：进行任务处理时，需要对涉及到的相关数据所在的网络位置进行定位，这就需要对分布在网络中的不同数据库服务器进行有效的管理。建立其元数据库，元数据信息包含数据库中的数据信息描述以及数据库服务器的网络位置等。

(3)DBMS Agent：是长期驻留在数据库系统上的 Agent，承担着现有数据库和移动 Agent 间的通信工作。DBMS Agent 一般不直接面向对用户的服务，在很大程度上协助移动 Agent 完成复杂的数据处理功能。因此，在移动 Agent 与 DBMS Agent 间留有彼此交互的专门接口。

3.2　服务层

移动 Agent 服务设施(Mobile Agent Environment, MAE)负责为移动 Agent 建立安全、正确的运行环境，为移动 Agent 提供最基本的服务(包括创建、传输、执行)，实施针对具体移动 Agent 的约束机制、容错策略、安全控制和通信机制等。

在通常情况下，一个 MAE 只位于网络中的一台主机上，但如果主机间是以高速网络进行互联的话，一个 MAE 也可以跨越多台主机而不影响整个系统的运行效率。MAE 利用 Agent 传输协议(Agent Transfer Protocol, ATP)实现 Agent 在主机间的移动，并为其分配执行环境和服务接口。移动 Agent 在 MAE 中执行，通过 Agent 通信语言(Agent Communication Language, ACL)相互通信并访问 MAE 提供的各种服务。

3.3　管理层

3.3.1　子任务时间表

对同一数据集的同一操作视为相同的子任务，记录其执行时间。多次执行后取其平均时间，根据子任务的执行时间可以根据给定的算法计算出子任务的最优执行序列。

3.3.2　协作管理器

在每一台服务器上都存在一个协作管理器，其功能包括：接收用户发送的任务并对任务进行划分，得到相应的子任务；控制任务执行过程中负责执行子任务的移动 Agent 之间的协作。协作管理器的相关模块介绍如下：

(1)任务分解 Agent：负责将接收到的任务进行划分，得到若干个子任务。并根据给定的算法得出子任务的最优执行序列，每一个子任务由一个 Agent 负责执行。将协同环境及其中的 Agent 集合 A 视为一个系统，A 上所有原子关系集合，称为系统协同关系，表示为

$$\boldsymbol{RS}=A\times A=\{R_{ij}\mid(\forall\ i\in A)\wedge(\forall\ j\in A)\}$$

其中，R_{ij} 描述 Agent$i$ 和 Agent$j$ 之间的关系；$\boldsymbol{RS}$ 是 Agent 的关系矩阵，表示为

$$\boldsymbol{RS}=A\times A=\begin{bmatrix}R_{11} & R_{12} & \cdots & R_{1n}\\ R_{21} & R_{22} & \cdots & R_{2n}\\ \vdots & \vdots & & \vdots\\ R_{n1} & R_{n2} & \cdots & R_{nn}\end{bmatrix}$$

系统协同关系描述了系统整体的协同状态，即系统中所有 Agent 协同关系的状况。在系统协同关系矩阵中，行列上 Agent 的顺序都与子任务最优执行序列上的具体子任务相对应。如 $R_{11}, R_{12},\cdots,R_{1n}$ 代表在子任务执行序列中完成第 1 个子任务的 Agent 到系统中完成其他子任务的 Agent 的协同关系。R_{ii} 是指 Agent 与其自身的关系，这对协同工作来说是毫无意义的，因为协同只有在 2 个或 2 个以上的 Agent 间才会发生。在本模型中各个 Agent 间的原子关系多为依赖关系和空关系。

(2)子任务队列：存放任务执行时的子任务序列。子任务序列是通过给定算法得到的最优子任务执行序列。子任务一旦被执行即执行出队操作。子任务的执行由子任务管理 Agent 来进行控制。

(3)子任务管理表：子任务管理表负责维护子任务的执行状态，在正常情况下包括正在执行，未被执行以及已被执行 3 种状态，当一个任务中的所有子任务的状态都是已被执行时，此任务即被完成。通过子任务管理表可以清楚地了解到子任务的执行状态。

子任务管理表的数据结构定义如下：

1)SubTaskID：子任务的标识符，唯一值；

2)SubTaskSta：子任务的状态；

3)SubTaskOper：子任务执行的操作类型；

4)SubTaskDataset：处理子任务涉及的数据集；

5)InputPara：输入参数；

6)InputParaType：输入参数类型；

7)STime：子任务执行的开始时间；

8)FTime：子任务执行的结束时间；

9)Etime：子任务的执行时间。

(4)Agent 管理器：在 Agent 管理器中将维护一个 Agent 管理表，不仅将负责完成的子任务与 Agent 进行一一对应，还用来作为 Agent Name 与网络实体位置的对照，只需要知道 Agent 的名称就可以从表中找到其网络位置。当创建一个新的 Agent 时，必须注册到表中，便于管理和维护。

(5)子任务管理 Agent：在任务执行的过程中，网络中可能存在多个移动 Agent 来处理同一个任务，多个 Agent 之间就会涉及到协作的问题。如果 2 个移动 Agent 之间需要进行协作，就必须要进行通信。这样就需要有一个子任务管理 Agent 对网络中负责执行子任务的 Agent 进行统一管理[2-3]。

当一个 Agent 完成子任务的时候，它向子任务管理 Agent 报告它当前任务已经完成，由子任务管理 Agent 根据协同算法通过 Agent 协同关系矩阵来决定它是将结果返回，还是直

接携带结果迁移到其他网络位置来与其他 Agent 协作完成处理任务。在 Agent 协同关系矩阵中，保存了执行各个子任务 Agent 之间的关系。子任务的协同算法流程如图 2 所示。

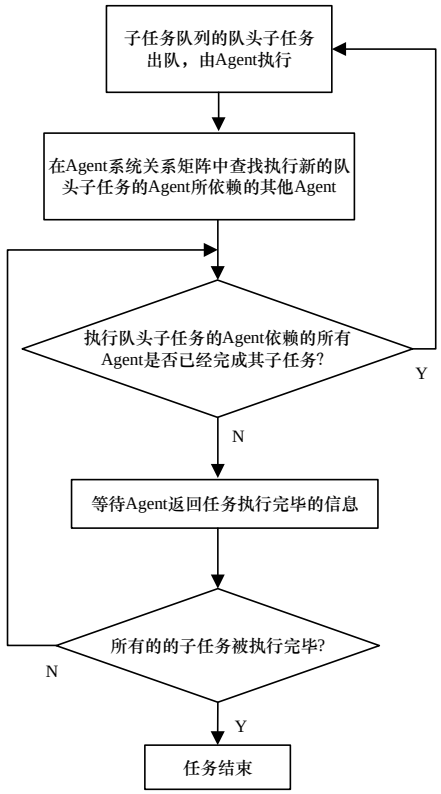

图 2　子任务的协同算法流程

4　子任务执行序列的建立

当任务管理器 Agent 把接收到的任务分解成多个子任务时，如何对这些子任务进行调度就成为任务处理效率高低的关键问题。最简单的方法是子任务之间用 AOV 网络来进行表示，子任务有序图如图 3 所示，其中，顶点表示子任务；有向边表示子任务间的依赖关系；顶点权值表示子任务的执行时间。

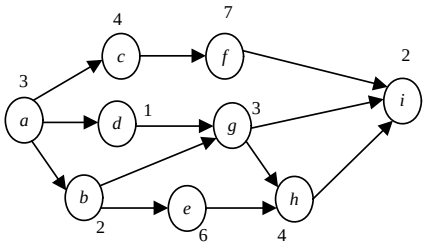

图 3　子任务有序图

对图 3 进行拓扑排序，可以得到多个调度序列，这样一个简单的 AOV 网络就可以得到 2 个或更多的序列。对于复杂的包括更多子任务的任务来说，它将会有更多处理子任务的序列，如果只是随机地选择一个序列来进行处理的话，将不能保证按照最优序列来处理任务。因此，在 MACPM 中，应用了文献[4]中改进的 Coffman-Graham 并行算法[4]。应用该算法必须了解每一个子任务的执行时间。用户提交的任务会被分解为若干个子任务，在协作管理器中包含一个子任务时间表，在子任务时间表中，将视对同一个数据库中的数据执行的同一操作为相同的子任务，记录其平均执行时间以及执行次数。在某个子任务被第一次执行的时候，将给定一个初始的执行时间值以便于在算法中计算。当这个子任务被执行多次后，它的执行时间将取多次的算术平均值，使这个数据能更准确地应用到算法中，保证得到的子任务序列更加准确。

对于给定的任务，根据 Coffman-Graham 并行算法得出一个较优的子任务执行序列，可以提高任务的处理效率。根据改进的 Coffman-Graham 并行算法，可以得到一个子任务序列 *a, c, b, e, f, d, g, h, i*，与前面通过拓扑排序得到的 *a, c, f, b, e, d, g, i, h* 序列相比较，可以得到如图 4 所示的调度结果。

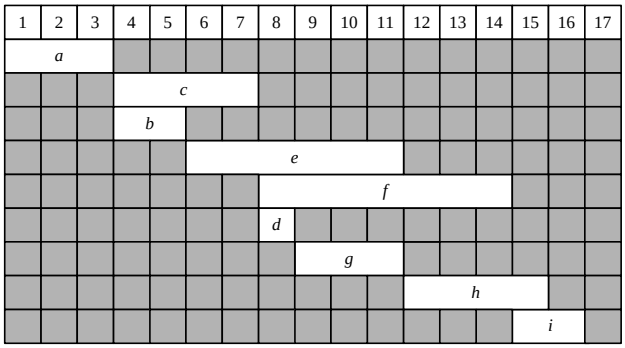

图 4　改进 Coffman-Graham 算法的调度结果

可以看出，当给定一个任务图时，AOV 网络的拓扑排序方法只能保证得到一个有效的序列，并不能保证此序列是最优的。通过改进的 Coffman-Graham 算法，可以得到一个最优的子任务调度序列。最优子任务调度序列可以保证总的执行时间最短。将改进的 Coffman-Graham 算法应用到本文提出的多 Agent 协同处理模型中，在很大程度上改进了对多个子任务的处理过程，提高了任务的处理效率。

5　结束语

本文根据移动 Agent 的特点，采用有效的并行处理算法，提出多 Agent 协同处理模型，建立了模型的体系结构。该多 Agent 协同处理模型可以用处理功能网络迁移的模式代替大量数据的迁移，在遇到复杂任务时利用 Agent 以最优的子任务执行序列来协同处理，从而大大提高了任务处理效率。该模型可应用于数字图书馆、数字城市以及数字地球等涉及数据量大、任务复杂的领域，具有实际应用价值。

参考文献

[1] 张云勇，刘锦德. 移动 Agent 技术[M]. 北京: 清华大学出版社, 2003.

[2] Wan Kaiyu, Alagar V, Yang Zongyuan. Trustable Ad Hoc Networks of Agent Societies[C]//Proc. of the 8th ACIS International Conf. on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing. Qingdao, China: IEEE Computer Society, 2007.

[3] Wang Yinghong, Keh Huan-Chao, Hu Tsang-Ching, et al. A Hierarchical Dynamic Monitoring Mechanism for Mobile Agent Location[C]//Proc. of AINA'05. Taipei, China: [s. n.], 2005.

[4] 张茂员，卢正鼎. 一种 Agent 数据库系统框架及其规则并行算法[J]. 软件学报, 2004, 15(8): 1157-1164.

编辑　顾姣健