第33卷 第20期 计 算 机 工 程 2007年10月
Vol.33 No.20 Computer Engineering October 2007
·软件技术与数据库· 文章编号：1000—3428(2007)20—0072—03 文献标识码：A 中图分类号：TP393

基于 Multi-agent 的轻量级工作流系统

刘 菲，侯海文
（山东大学计算机科学与技术学院，济南 250061）

摘 要：讨论了采用 multi-agent 技术和轻量级理念设计的工作流系统，并给出了工作流系统结构的具体表示、任务分解和资源分配的方法。提出了一种在多 Agent 中基于 ontology 信息的传递方法，减轻了工作流中通信的负担，并解析了该系统框架的功能，说明了系统的优势，介绍了系统的实现方案。

关键词：multi-agent；轻量级；工作流系统；ontology

Lightweight Workflow System Based on Multi-agent Technology

LIU Fei, HOU Hai-wen
(School of Computer Science and Technology, Shandong University, Jinan 250061)

【Abstract】 This paper discusses workflow system based on multi-agent and lightweight design idea, proposes the representation of the way to implement the workflow system, methods of task decomposition and resource assignation. A method based on ontology is proposed and discussed for information transfer among agents in workflow system. The method lessens the communication burden of the workflow system, analyzes function of this architecture, shows the superiority of the model, and introduces implementation scheme.

【Key words】 multi-agent; lightweight; workflow system; ontology

1 概述

由于不同领域的Agent之间繁琐的通信，因此具有自组织、自学习和协同工作能力的轻量级工作流系统[1]将是未来的发展趋势。

轻量级工作流系统从“够用”、灵活和低成本的原则出发，不追求工作流引擎功能的完备和复杂，实现工作流管理系统中必不可少的功能和特征，并构造出各种灵活的工作流应用系统。Tagg[2]等学者在对工作流引擎的描述中，曾经使用过“轻量级”这一术语，但其侧重点在于如何构造一个“瘦客户端”。本文的侧重点则是要设计一个充分支持工作流特征的小型内核，它可以无缝地嵌入到传统的应用开发环境中。

在轻量级工作流系统中，面临的两个主要问题是任务分解和资源分配，着眼于流程信息的驱动和管理，能够协调企事业单位内的各种资源，弥补传统的管理信息系统只对独立的资源进行维护和处理的弊端，并通过工作流引擎的驱动使任务流程中各个具有一定关系的任务在一定程度上自动运行，从而高效、灵活地达到企事业单位的业务需求。

许多文献已经提出了使用 Agents 设计和开发工作流系统，文献[3]提出利用 Agents 实现工作流定制服务；文献[4~5]提出使用 Agents 协调 Web 服务来实现工作流。

笔者提出的轻量级工作流系统不同之处在于：(1)过程模型能够动态地改变；(2)提出新的分解任务执行的方法，在单独任务中可以动态地调用合适的 Web Service 服务；(3)在运行期间，可以根据历史数据动态选择资源。

笔者结合 multi-agent 和 ontology 技术提出轻量级工作流系统架构，研究了任务分解和资源分配问题。该架构实现了工作流管理联盟中必不可少的核心功能，集成现有的一些工作流管理系统的优点，在此基础上构建并实现了一些新的符合用户需求的功能。

2 基于 multi-agent 的轻量级工作流系统

2.1 工作流系统结构

基于多 Agent 的轻量级工作流系统结构主要由 7 个 Agent 组成，提供了基本的控制工作流的作用，如图 1 所示。其中，UDDI server 用来调用外部 Web Service。工作流的执行是通过系统中的 Agents 来实现的。

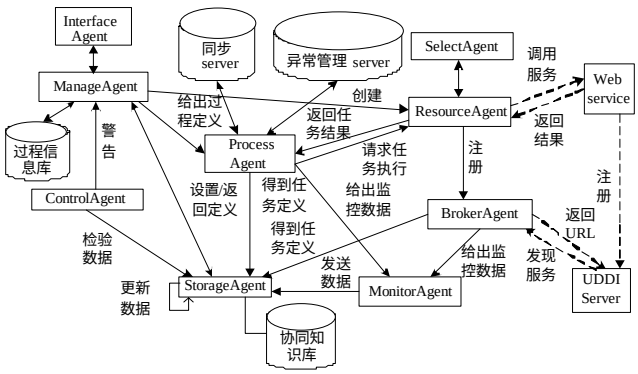

图 1 基于 multi-agent 的工作流系统结构

InterfaceAgent 代替用户发出过程启动请求；代替用户自动处理消息或等待用户处理，并将处理过程存入内部的知识库中，为用户提供处理建议。ManageAgent 是整个系统的核心，其功能和地位类似于工作流参考模型中的工作流引擎。主要提供基本的工作流管理者所需要的功能，例如任务的创建和删除、过程定义、新的过程实例的初始化和 ResourceAgent 的创建。此外，还能够根据自身功能逻辑、当

作者简介：刘 菲(1978－)，女，博士研究生，主研方向：移动计算，智能计算；侯海文，硕士

收稿日期：2006-11-30 **E-mail：**lflemmon@163.com

前运行负载等情况，将过程定义和过程实例运行时的信息在不同的 ManageAgent 之间传递。当业务过程中的一个事务被启动，ManageAgent 同时启动一个 ProcessAgent，并给该 Agent 设定运行所需要的参数。ProcessAgent 验证过程模型并执行。对每一个工作流实例，由 ProcessAgent 分配任务给 ResourceAgent。系统中每个资源都由自己的 ResourceAgent，并且在至少一个 BrokerAgent 处注册，由 BrokerAgent 分配资源给过程。StorageAgent 管理所需的永久数据，例如任务、角色和过程的定义以及所要监控的数据。MonitorAgent 主要负责对过程实例运行状况的监视和管理功能，它通过对 ProcessAgent 和 BrokerAgent 给出的监控数据进行查询和监视，得到活动的执行信息，如活动执行是否已经超时、活动执行是否失败等，然后根据这些信息通过接口向 StorageAgent 发送。ControlAgent 不断观察一些异常情况，并发送警告信息给 ManageAgent。由 ManageAgent 决定采取适当的解决办法来执行。SelectAgent 接收 RsourceAgent 传来的有关下一步任务的信息，选择最适合执行此任务的节点，传递该节点的相关信息至 RsourceAgent。RsourceAgent 携带着初始的路由信息，通过自身对过程定义的解释执行，并逐步迁移来完成各个活动的调度。当 RsourceAgent 从一个节点到达另外一个节点时，必须和该节点的 InterfaceAgent 进行交互，同时把任务转交给该 InterfaceAgent，完成制定的任务后，由 InterfaceAgent 将结果返回给 RsourceAgent，该 RsourceAgent 通过复制将其传递到下一个节点，直到遍历完所有的节点。

同步 server 用来维护各个位置的属性，并且存放了各个工作流的过程定义、状态等信息，只有获得权限的 ManageAgent 才能访问该流程的过程定义，并且具有编辑的权限，修改完毕后，同步 server 会将修改后的过程定义发送给该工作流的所有位置上的协同 Agent，使它们可以同步感知系统的更新，按照新的定义来执行工作流，从而使当前的修改能够“即时生效”，做到了动态的适应性柔性。此时，正处于任务执行状态的 ManageAgent 也可以通过与当前位置的 InterfaceAgent 的交互来获得最新的过程定义信息，使之保持同步。异常管理 server 接收并处理工作流执行过程中产生的异常或中断，保证工作流在发生意外的情况下能够及时处理异常并继续正常工作。

2.2 任务分解和资源分配

多 Agent 接收任务后，首先将该任务分解成多个子任务，并将这些子任务分别分配给系统中的执行单元 Agent 去执行。接收到任务的这些 Agent 在协调者的管理下相互合作、共享系统资源，完成各自的任务。当所有这些子任务完成后，协调者将这些子任务的结果进行集成，形成整个任务的结果。任务分解执行的启发式算法如下:

(1)必须确定一组满足任务的约束条件的操作；

(2)将在前一步中确定的操作分配给执行单元执行，并使得耗费的执行开销和通信开销最小。

分解任务以后，ProessAgent 要执行一个工作流模型定义，就要给过程模型定义中的活动分配具体的执行者。在过程的执行过程中动态地为每个任务分配需要的资源。这就要求 ProessAgent 在执行过程中动态地请求 BrokerAgent，寻找合适的资源信息，然后分配给任务。

WorkaroundAgent 就是一个给用户提供过程描述界面(可以编辑过程定义、监视过程执行进度的界面)的 Agent。在开始执行任务前，由 ManageAgent 创建出与参与整个过程节点数相等数目的 WorkaroundAgent，按照位置信息逐个派遣 WorkaroundAgent。每个节点只有在获得了 WorkaroundAgent 后才可能看到工作流的执行情况：

(1)BrokerAgent 请求 ResourceAgent 搜寻能够执行该任务的资源的历史数据。

(2)根据这些数据，BrokerAgent 选择一个能够执行任务的最好的资源。如果 BrokerAgent 不能够分配资源，那么它发送失败消息给 ProessAgent。ProessAgent 将自行搜索合适的资源，并完成任务。

整个过程如图 2 所示。

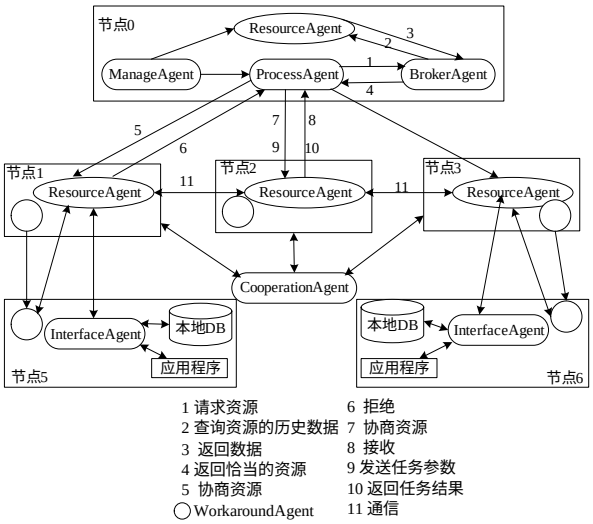

图 2 分配资源给任务的序列

2.3 基于 ontology 的信息交换

在多 Agent 系统中，由于通信延迟或者某一个关键 Agent 的故障的影响，使得信息接收的可靠性及稳定性成为负担。Agent 之间的通信可以通过 ontology 来实现。当 Agent 需要彼此交互时，它们可以使用 Agent 通信语言 KQML(knowledge query and manipulation)。KQML 提供了一个 Agent 交换信息和知识的架构。其核心是一套可扩展的表述行为集，其中包含了希望接收方执行的一些动作，它为 Agent 定义了一些允许其访问彼此知识和目标库的操作，这些行为构成了 Agent 之间相互通信的基础。

图 3 显示了基于 ontology 的多 Agent 的信息交换结构。

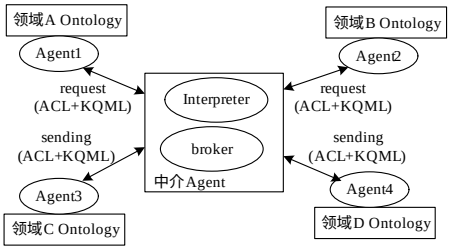

图 3 基于 ontology 的多 Agent 信息交换结构

Agent 常常代表系统中的不同角色，拥有单独的 ontology。这样，Agent 间就很难有效地相互通信。对于一个多 Agent 系统来说，最好使用公共的 ontology，每个 Agent 就能共享公共 ontology。中介 Agent 可以促使多 Agent 共享知识和信息，它的作用就好像 Agent 间的匹配器，提供一个公共的 ontology 来解释 Agent 的上下文，并且传送这些内容给其他 Agent。中介 Agent 是由两个组件构成：interpreter 和

broker。当中介 Agent 接收到一个请求，它的 broker 就会发送这个请求给 interpreter，根据转换规则，如本体间的匹配，interpreter 将该请求转换成目标 Agent 可以理解的形式发送过去。这样，就实现了多 Agent 间的信息交换。

3 系统的优势分析及实现方案设计

分析这个基于 multi-agent 的轻量级工作流模型，并与传统的工作流系统比较，它在下列几个问题的处理上体现了极大的优势：

(1)执行中的模式僵化问题。在工作流执行的过程中，工作流引擎被动地解释执行过程定义。对于有些特殊的业务流程，需要动态地修改工作流的执行路线。利用 Agent 技术可以提高工作流执行的灵活性，实现了流程的动态修改。也对异常情况和例外的处理提供了更优的方案。

(2)工作流系统中的资源冲突问题。因为工作流系统调用着企业内的各种资源，所以 2 个工作流有可能会竞争同一资源而发生冲突。利用多 Agent 资源分配和基于 ontology 的信息交换可以有效地解决这个问题。

(3)工作流任务的动态划分问题。该算法任务执行 Agent 将工作流任务动态划分，并依照协调信息发送到后续恰当的任务执行 Agent。

目前，笔者已经对“多 Agent 工作流系统”结构进行试验。使用一台 Dell2400 服务器作为 UDDI Server，3 个工作位置分别为 2 台 Dell2400、1 台 IBM5000。为了保证安全性，用 1 台 IBM4400 作为备份服务器来保证系统故障时继续工作。客户端采用 Windows 平台，每个节点均具备迁移工作流平台运行环境，Oracle 数据库。在设计和测试过程中，针对各种不同的情况均做了模拟测试，运行结果正确，成功地解决了实际问题，同时也达到了其轻量级的要求。

4 结束语

随着企业应用规模和功能的不断增大和调整，基于工作流技术的应用系统必将是软件技术的发展方向之一。该文通过分析当前多 Agent 工作流系统的一些不足之处，提出了一种基于 multi-agent 技术的轻量级工作流系统结构，研究了其中的关键算法和实现机制，重点阐明了多 Agent 的工作流程。根据 multi-agent 的工作流，描述了工作流系统的运行，使得工作流系统具有很大的柔性和动态性，根据 ontology，实现了多 Agent 的通信，使得系统具有支持分布异构环境和系统重构的能力。

未来的研究工作是：增加工作流的自学习功能，进一步完善基于 multi-agent 技术的轻量级工作流系统，提高系统性能分析和容错能力。

参考文献

1 Muth P, Weissenfels J, Gillmann M. Integrating Light-weight Workflow Management Systems Within Existing Business Environments[C]//Proceedings of the 15th International Conference on Data Engineering. 1999.

2 Tagg R. Preliminary Design of a Lightweight Workflow Server[C]//Proc. of the 8th Australasian Conf. on Information Syst., Australia. 1997.

3 Zeng L, Ngu A, Benatallah B. An Agent-based Approach for Supporting Cross-enterprise Workflows[C]//Proc. of the 12th Australasian Database Conference. 2001: 123-130.

4 Blake B. An Agent-based Cross-organizational Workflow Architecture in Support of Web Services[C]//Proc. of the 11th IEEE International Workshops on Enabling Technologies. 2002: 176-181.

5 Vidal J, Buhler P. Multi-agent Systems with Workflows[J]. IEEE Int'l Journal on Internet Computing, 2004, 8(1): 76-82.

(上接第 71 页)

5 性能分析

本文提出的挖掘方法，可以找出所有满足约束条件的频繁项目集，并能有效地减少短模式的生成。同已有的支持度降低的关联规则挖掘方法相比，该方法针对长度支持度降低的约束条件，结合相应的性质 4、性质 5，在频集生成过程中，只需找到满足长度条件的项目集进行组合，因此可有效地降低大量候选项目集的生成。与挖掘频繁项目集方法[7]相比，本方法可有效地降低算法的复杂度，不存在一次性放入内存等缺点。而且，采用向量法挖掘所有的频繁项目集，只需扫描一遍数据库，其后频集的计算只需找到相应的项目集进行“与”运算[8]即可，因此，提高了挖掘效率。

6 结束语

本文提出了新的长度支持度约束条件，在该约束条件下，不仅可以获得具有潜在价值的长模式，也可以有效去除许多无用的短模式。而且，可以根据不同的数据要求进行参数调整，具有一定的灵活性。利用向量法来挖掘长度支持度降低的关联规则，结合相应的性质 4、性质 5，既有效地减少了大量候选项目集的生成，也提高了挖掘的效率。因此，本文提出的挖掘方法，具有一定的理论及实际价值。

参考文献

1 Agrawal R, Srikant R. Fast Algorithms for Mining Association Rules[C]//Proceedings of the 20th Int'l Conference on Very Large Data Bases, Santiago, Chile. 1994.

2 Han J, Pei J, Yin Y. Mining Frequent Patterns Without Candidate Generation[C]//Proc. of SIGMOD Int'l Conf. on Mangement of Data. 2000.

3 Park J S, Chen M S, Yu P S. An Effective Hash-based Algorithm for Mining Association Rules[C]//Proceedings of the ACM SIGMOD International Conference on Management of Data. 1995: 175-186.

4 Burdick D, Calimlim M, Gehrke J. MAFIA: A Maximal Frequent Itemset Algorithm for Transactional Databases[C]//Proceedings of the 17th Int'l Conf. on Data Engineering. 2001.

5 Lucchese C, Orlando S, Perego R. Fast and Memory Efficient Mining of Frequent Closed Itemsets[C]//Proc. of the 2nd IEEE ICDM Workshop on Frequent Itemset Mining Implementations. 2004.

6 Pasquier N, Bastide Y, Taouil R, et al. Discovering Frequent Closed Itemsets for Association Rules[C]//Proc. of the 7th Int'l Conf. on Data Theory. 1999.

7 Seno M, Karypis G. LPMiner: An Algorithm for Finding Frequent Itemsets Using Length-decreasing Support Constraint[C]//Proc. of the 1st IEEE Data Mining, San Jose, USA. 2001.

8 李 哲, 杨兆中, 庞炳章. 大型数据库中关联规则的向量法挖掘[J]. 计算机工程, 2003, 29(8): 97-99.