计算机科学 1995Vol. 22№. 3

一种产生式系统的并行推理模型*)

廖明宏 郭福顺 程退安
(哈尔滨工业大学计算机科学系 哈尔滨 150001)

摘 要 The production system is an effective method for knowledge representation. With the scale of its knowledgebase become larger, its inference efficiency turn lower. This paper analyzes the inference mechanism of the production system, and designs a partition algorithm and a parallel inference model called GPIM for the parallel processing of the production system on the architecture of the distributed multi-computer systems, they are more suited to the processing of large scale knowledgebase.

关键词 Production system, Parallel processing, Partition algorithm, Parallel inference model.

1. 引 言

在人工智能领域中，产生式系统是一种比较有效的知识表示方法，并得到了广泛的应用。目前许多较为成功的专家系统都是用产生式系统实现的[1]，随着应用领域的不断扩大，其知识库的规模逐渐增大，而它的推理效率等性能指标随着知识规模的增大而下降，从而影响了它在智能系统中的应用。

并行分布式计算机系统具有高度并行和分布控制的特点[2]，是提高计算机运行效率的有效途径[3]，也是提高产生式系统推理效率的有效平台。为此必须首先对产生式系统中的并行任务进行识别和划分，以便在多处理器或多机系统环境下实现分布式的并行推理。

分布式机系统具有资源共享和分布处理优点，提供了实现产生式系统并行处理的平台。本文首先分析产生式系统的推理机制，并讨论支持产生式系统并行推理的硬件环境，然后提出并行任务的识别与划分算法，和并行推理模型的基本思想。

2. 产生式系统语言 OPS5

一个产生式系统由一个规则库，一个数据库和一个推理机三部分组成。其中，规则库存放具有"IF …THEN"形式的规则知识；数据库存放推理中的事实数据，推理机用规则库和数据库进行推理。

OPS5 语言是产生式系统语言的代表。在 OPS5 中规则库称为产生式存贮器(PM)，它用于存放一组产生式规则。规则的形式为：

(P〈规则名〉〈LHS〉→〈RHS〉)

其中：〈LHS〉是规则的左手部，表示条件。〈RHS〉是规则的右手部，表示结论。

数据库在 OPS5 中称为工作存贮器(WM)，用以存放在推理过程中产生的当前数据元素：工作存贮元素(WME)，WME 的形式为：

(〈类名〉↑〈属性〉〈属性值〉… ↑〈属性〉〈属性值〉)。

OPS5 的推理机控制整个产生式系统的执行过程，它由匹配-冲突归结-点火这三个阶段循环组成。

阶段 1——匹配：把产生式系统每个条件元素与 WM 中的每个元素进行比较，看哪个可与条件元素匹配，如果至少有一个 WME 与之匹配，则该条件被满足；如果产生式 LHS 所有的条件都被满足，则称该产生式被满足，匹配过程的输出是一个冲突集，冲突集的对象叫做示例。

阶段 2——冲突归结：根据一定的策略从冲突集中选择一个示例，而这个示例将在点火阶段被执行。若选择不出一个示例，则推理过程结束。

阶段 3——点火：选中的产生式的 RHS 动作被依次执行。

重复执行上述三个阶段过程。

在这个推理过程中，匹配阶段最费时间，为此，C. Forgy 提出一种有效的匹配算法：RETE 算法[4]。它的基本思想就是根据所有产生式规则左手部条件，构造一个数据驱动的识别网络(称为 RETE 网)，匹配时将每个 WME 从 RETE 网的根结点出

*)该项研究属国家自然科学基金和航天基金资助项目。廖明宏 博士，研究方向人工智能，并行处理。郭福顺 教授，研究方向人工智能，并行处理，操作系统。程退安 教授，研究方向智能机体系结构。

发向下流动匹配，网的叶结点是匹配成功的冲突集。由于 RETE 网采取存贮中间匹配结果和共享匹配操作等办法，提高了匹配阶段的进行速度。因此 RETE 算法成为产生式系统的基本匹配算法。

RETE 算法并不能根本解决产生式系统推理效率低的问题。为此又出现了许多并行匹配算法和其它并行处理技术。在这些并行处理方法中，多规则点火系统[5]是一种较有前途的做法。它的基本思想是在产生式系统的一个推理周期中，尽可能多地点火多条规则，从而使整个推理周期的三个阶段都并行处理，以此来提高系统的推理效率。但为保证多规则点火系统与顺序点火系统结果的等价性，系统必须解决相容性和汇流性(CONVERGENCE)问题。所谓相容性就是要求并行点火的规则必须是不相互矛盾的；而汇流性问题是对相容性条件的补充，它要求从全局上保证多规则点火系统的正确性。目前许多实验系统对这两个问题的解决方法各不相同，对系统推理效率的提高也有所不同，但到目前为止，还没找出一种公认的有效的办法。

本文旨在一种分布式多机系统上设计产生式系统的多规则点火系统，以期得到较好的推理效率。

3. 分布式硬件系统

在分布式多机系统中，网络的拓扑结构是整个系统设计的关键，为有效支持产生式系统的并行分布式处理，我们对多机系统中的通讯网络提出如下要求：

(1)网络通讯时间开销尽可能少；

(2)各处理机对总线的申请尽可能不相冲突；

(3)网络的设计应利于系统的可扩充性。

基于以上三点考虑，我们设计的分布式多机系统的结构如图 1 所示(以四台处理机为例)。

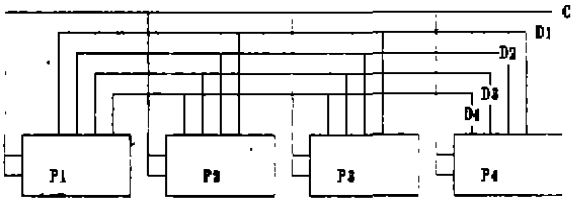

图 1　分布式多机系统的结构

说明：P1—P4 表示四台处理机；D1—D4 表示四条数据通讯总线；C 表示控制总线。

当某台处理机 Pi 欲与其它处理机通讯时，它从四条通讯总线上选择一条不忙的线，(假设为 Dj)进行发送信息，此时，其它处理机与 Dj 相连的端口为接收端，用以接收由 Pi 发出的信息。所有通讯总线都是双向的，它们的发送与接收是可变的。

由于四台处理机共有四条通讯总线，它们在通讯时不会发生申请总线冲突现象，此时的控制总线 C 所起的作用不大。但为了系统的可扩充性，当增加处理机而不增加通讯总线时，控制总线开始发挥作用。系统可以利用控制总线检测通讯总线的忙闲状态，以便决定是等待还是选择某条总线进行通讯。

这一分布式拓扑结构适于分布式多机环境上产生式系统的并行分布式处理。

4. 产生式系统的并行划分

为使产生式系统能在上一节介绍的多机环境中运行，有必要对一个产生式系统进行划分。具体地，把一个产生式系统分解成若干个相对独立的子系统，并分配到不同的处理机上。每个子系统由若干规则子模块及其相应的数据库组成。每台处理机对各自的子系统进行推理，所有的处理机用异步的工作方式运行，它们之间采用消息传递方式进行通讯，以此来协调求解同一个产生式系统。

系统的划分准则如下：

(1)相关程度大的规则尽量放在同一子系统中，以减少处理机之间的通讯开销。

(2)可并行的规则尽可能放在不同的子系统中，以提高处理机的利用。

(3)各子系统的工作量应尽可能均匀，以达到系统的负载平衡。

4.1　工作存贮器的划分

OPS5 的工作存贮器与关系数据库存在一种自然对应关系，见表 1。

表 1　工作存贮器与关系数据库的对应关系

工作存贮器	关系数据库
类	关　系
属性	域
工作存贮元素	记录

由表 1 可以看出，OPS5 中的一个类对应关系数据库中的一个关系，因此，可以为每一个类建立一个关系数据库，数据库的结构由类的属性域来确定，把工作存贮器元素按它们的类划分成不同的数据库，使得在匹配时，每条规则只关心与之有关的数据库记录而不必理会其它数据库上的记录。

4.2 规则库的划分

规则库的划分就是根据系统的划分准则将规则库划分成若干相对独立的子模块。规则库的划分算法很多，这里介绍我们提出的一种更为有效的划分算法：按规则相关度划分算法 DDPA。

假设，RB：表示规则库；Ri，Rj∈RB；MSET：表示模块集；Mi，Mj∈MSET；R0：表示大于等于 0 的实数集。则有如下定义：

定义 1（规则间的相关函数） 对一给定的规则库 RB，可定义一个函数 F，使得对 RB 中任意两条规则 Ri 和 Rj，有 F(Ri，Rj)∈R，则称函数 F 为规则间的相关函数。

定义 2（模块间的相关函数） 对一给定的模块集 MSET，可定义一个函数 F，使得对 MSET 中任意两个模块 Mi 和 Mj，有 F(Mi，Mj)∈R0，则称函数 F 为模块间的相关函数。

相关函数可以用来刻画规则（或模块）之间的相关程度（简称相关度），可以对它赋上大于等于 0 的实数值，这个值越大，表示规则（或模块）之间的相关程度也越大。对一给定的规则库进行划分，其划分结果不唯一，一般可根据系统中的处理机台数，给出欲划分的模块数。

DDPA 算法的基本思想是：

S1[初始化]：把规则库中每条规则看成一个模块；

S2[合并冲突规则]：把每对相互冲突的规则合并到同一模块；

S3[按规则相关度]：若 S2 没达到给定模块数的要求，则对其它规则按其规则相关度的大小由大到小进行两两模块合并；

S4[按模块相关度]：若 S3 没达到给定模块数的要求，则按模块间的相关度大小由大到小进行两两模块合并，直到满足给定模块数的要求。

5. 产生式系统的并行推理

为使产生式系统在分布式环境中进行并行推理，我们提出一种新的并行推理模型：目标驱动的并行推理模型 GPIM。它是一种正反向推理相结合的多规则点火系统。GPIM 的基本思想是：

(1)在多机系统中，指定任意一台处理机为主机；其它处理机为从机。开始时，规则库和数据库存放在主机上；

(2)主机把规则库中的子模块及其相应的数据库分布到其它处理机上；

(3)主机把规则库中给出的推理目标与规则库中所有规则的右手部进行比较，找到与目标相匹配的所有规则。把这些规则的左手部作为新的子目标，以广播方式发给其它处理机（这一工作类似反向推理）。

(4)从机收到与之有关的子目标，开始进行正向推理。其推理算法采用 RETE 算法。

(5)从机之间采用异步工作方式，所有的处理机同时进行各自的推理工作，并采用消息传递方式进行通讯。即当某一处理机点火一条规则并产生对公共数据库（被两个以上模块所引用的数据库）的操作（插入、删除或修改等），它首先修改本单元上保存的数据库，然后把修改信息发送到引用该数据库的所有处理机上。接收端上的处理机暂时中断推理工作，接收发来的消息，并做相应的修改、然后继续推理。

(6)当某一从机推导出一个子目标，它把该目标返回给主机，然后继续它的正向推理工作。

(7)当某一从机推理结束，它返回给主机一个等待信息；然后等待其它处理机发给它新的对数据库操作的信息，或由主机发给它的控制信息。若其它处理机促使它继续工作，它首先返回给主机一个忙信息，然后继续推理。

(8)主机收到从机发来的子目标时，将它与目标规则的左手部进行匹配，若目标规则满足，则点火该目标规则。然后通知从机，终止各自的推理工作，最后结束整个系统推理。若目标规则没被满足，主机继续等待从机返回的子目标。

(9)若所有从机都返回给主机等待信息，且此时目标规则还没被满足，主机返回给用户目标没被满足的信息，然后拷贝回从机上最新的规则库和数据库，并结束整个推理过程。

从以上 GPIM 的基本思想可以看出，它能够组成产生式系统的各个子系统，分布到不同的处理机上，并协调各处理机进行并行推理，以达到对整个系统推理的目的。

GPIM 模型可以简单有效地解决相容性和汇流性问题。其中，相容性问题可以在规则划分过程中得到解决，因为规则划分算法首先将相互冲突的规则划分到同一模块，它们被分布到同一处理机上，从而避免相互冲突规则同时点火的现象。汇流性问题的解决很简单，它只要求用户给出推理目标，GPIM 就可以根据推理目标引导整个推理过程沿着推理目标方向进行，从而保证推理结果的正确性。

（下转第 44 页）

左右。

(2)只构造从 GB2312 标准到 CJK 汉字的转换表 A，这个转换表用线性数组实现，并可完成从一个汉字的 GB2312 编码到 UCS 编码的转换。从 CJK 汉字到 GB2312 编码的转换通过查找表 A 中的元素并计算该元素在表中的偏移来实现。当表 A 按 GB2312 标准的编码次序组织时，表 A 的内容(对应的汉字 UCS 编码)是无序的，可采用顺序查找方法。

假设每个汉字出现的频率相同，则成功查找的平均查找长度为：(1＋2＋…＋6763)/6763＝3382，不成功查找的查找长度为 6763。

这种方法将进行多次比较操作，转换速度最慢，但占用最少的内存空间，只需要 13K 左右。在内存要求苛刻的情况下可以考虑采用。

(3)做为方法一和方法二的折中，首先构造从 GB2312 标准到 CJK 汉字的转换表 A，这个转换表用线性数组实现，并可完成从一个汉字的 GB2312 编码到 UCS 编码的转换。然后构造一个表 B，表 B 的元素为一个指向表 A 的一个元素的指针，元素的组织次序可有多种方式：

①元素按所指向的表 A 元素的值(一个汉字的 UCS 编码)的从小到大次序(也可按相反次序)排列，从 CJK 汉字到 GB2312 编码的转换可通过对表 B 的元素进行一次间址后的有序表二分查找并计算在表 A 中的偏移得到。这种方式下一次成功查找的最大查找次数为：$[\log_2 N]=[\log_2 6768]=13$，不成功查找的最大查找次数为：$[\log_2 N]+1=[\log_2 6763]+1=14$。

②元素按所指向的表 A 元素的使用频率的高低次序排列，从 CJK 汉字到 GB2312 编码的转换可通过对表 B 的元素进行一次间址后的表顺序查找得到并计算在表 A 中的偏移得到。假设 GB2312 标准中按区位码次序的汉字的使用频率分别是 p1，p2，…，p6763，那么，成功查找的平均查找长度为：p1＋2×p2＋…＋6763×p6763，其中，p1|p2|…|p6763＝1。并且有 0<p6763≤…≤p2≤p1<1。不成功查找的查找长度为 6763。

这种方法将进行几次比较操作和一次间址操作，其转换速度低于方法一，但高于方法二；内存空间的占用量也居前两者之间，约需要 27K。

经我们统计，在这两个标准中，恰好具有相同排列次序的汉字数目不超过 9 个。因此，如果构造分段的转换表性能不一定能有很大改观。另外，还可以构造哈希函数实现转换。总之，采用什么方法实现两种标准间的转换应根据实际情况而定。

4. XDUCS Windows 的体系结构

我们以应用程序形式开发并运行，其体系结构如图 4 所示。

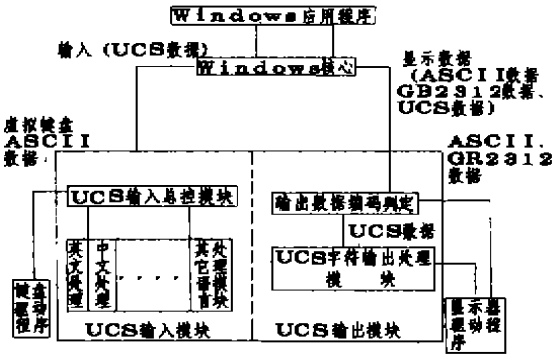

图 4　XDUCS Windows 体系结构

XDUCS Windows 的主要工作是支持 UCS 字符的输入和显示，因此，系统主要由输入、输出两个大模块组成。这两大模块位于 Windows 核心和设备驱动程序之间。输入模块根据当前用户选择的语言，将键盘输入或者键盘输入序列转换成为一个两字节的 UCS 编码，并形成两个字符消息发送给 Windows 或应用程序。显示输出模块首先判断所要显示数据采用的编码形式，对于采用 UCS 编码的数据，由 UCS 字符输出模块处理，对非 UCS 数据则调用 Windows 原来的字符输出函数将其输出。(未完待续)

(上接第 39 页)

参考文献

[1] Anoop Gupta, Parallelism in Production Systems, Morgan Kaufmann Publishers, Inc. California, 1987

[2] 孙钟秀，谢立，90 年代的分布式计算机技术，计算机学报，第 3 期，1992

[3] 祝旺发，分布式人工智能，计算机研究与发展，第 10 期，1990

[4] C. L. Forgy, Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, Artificial Intelligence 19, Sept. 1982

[5] Steve Kuo and Dan Moldovan, The State of Art in Parallel Production Systems, Journal of Parallel and Distributed Computing 15, 1992