计算机科学1997 Vol.21 №.6

# 基于曙光并行机的客户/服务器结构的数据开采*)

Client/Server Architecture for Data Mining Based on Parallel Computer Dawning-1

黄刘生 刘 清 陈国良

(中国科学技术大学计算机系 中国合肥高性能计算中心 合肥230027)

**摘 要** The rapid accumulation of the data in the real world demands for the development of powerful knowledge discovering tools. Echoing this demand, knowledge discovery in databases (KDD), focusing on the discovery of knowledge out of myriad data, develops corresponding techniques and tools. In this paper, we propose a client-server architecture consisting of a data mining client, organizing and controlling the data processing, as well as a data processing server in charge of parallel data handling.

**关键词** Data mining, Knowledge Discovery in Databases (KDD)

## 一、引言

KDD是指从数据库中发现有价值信息的整个过程,而数据开采(DM)则是指在这整个过程中的一个步骤,是针对某一具体问题的算法的实现。除了DM步骤,KDD还包括前处理(如数据准备等)和后处理(如采集到信息的形式化表示等)[1]。

在实际应用中,DM中最主要的处理操作集中在数据库中的资料处理,DM最主要的困难就是大规模的实时数据得不到行之有效的处理[4]。因此,我们提出一种客户/服务器结构,将数据库资料处理任务独立出来,由并行计算机来完成;而把DM中的其它任务仍交给客户端完成。我们的数据处理服务器是在对称紧耦合共享存储、多处理机的曙光一号上实现的。

下面首先介绍DM的有关概念以及我们的数据开采对象,然后详细说明我们的数据开采客户的工作机制,接着通过一个例子来描述并行机上数据处理服务器的实际处理过程,最后给出的是一些实验结果。

## 二、数据开采

随着所需处理的实际问题不同,数据开采所采用的策略与算法也不尽相同[2]。事实上,没有一种行之有效的数据开采方法普适于各种类型的问题。我们这里的数据是来源于一家食品超级市场的销售数据。数据的存储采用关系数据库,包含两个关系模式,一是顾客信息数据库C(CNum, Date, Time, Total),其中CNum是顾客编号,Date是日期,Time是付款时间,Total是总金额;另一是出售商品数据库G(GNum, Name, Type, Amount, Price, CNum),其中GNum是商品编号,Name是商品名称,Type是商品类别,Amount是数量,Price是单价,CNum是购买本商品的顾客编号。由于超级市场的顾客流动量大,实时数据积累很快,平常的信息处理只是抽取其中极其少量的信息加以利用,如顾客总数、总销售金额等,而对货架上商品的摆放量,不同商品的合理布局等直接关系到商业利益的一些统筹安排都只是停留在管理人员的经验判断层次,缺乏深层次的分析,也往往跟不上客观环境的变化。

针对这样一类对客观实时关系型数据进行分析,从中找出不同关系的内在联系的一类问题,习惯上是采用迭代的方法:先按照一定方法推断出一些规则,对数据资料进行处理,筛选其中部分有价值的规则,由此再产生一些新的详细规则,以此类推,直到推断出可信的知识为止[3]。

对于一家食品超级市场的销售数据,人们希望从中能获得有关指导商家进货、方便顾客购物的一

*)中国科大恒星计算机技术研究所资助项目,同时部分受到国家863项目(863-306-ZD-07-1)的资助。
黄刘生 副教授,主要研究领域为数据库、并行分布式计算。刘清 硕士研究生,研究方向为并行分布式计算。陈国良 教授,博士生导师,研究领域为并行分布式计算和智能计算。

些有价值的知识，譬如去发现人们在不同的时间，采购的商品所具有的内在规律性；另外商场内商品货架的安排，以及商品数量的合理控制不但方便顾客，也减少工作人员的劳动量。这些极有价值的知识都是可以从销售数据库中发现的。

举例而言：我们要考察购买咖啡的一类顾客的购买行为，很容易发现他们伴随着要购买一些甜味佐料，如糖等。这样一来，把糖等甜味佐料商品陈放在咖啡附近，就方便了顾客的购物。

## 三、数据开采客户

数据开采客户负责生成规则，然后将规则分解成一些基本的数据库处理操作，并将这些操作提交给数据处理服务器来完成，利用反馈回来的信息，对规则进行筛选，如此反复迭代到一定深度为止，最后将所得到的规则用自然语言表示，即得到了所要求的知识。

### 1 规则的生成与筛选

所谓规则，就是事物间因果关系知识的形式化描述。事实上，人们发现的知识常常是一些概然的规律，因此在 DM 中，一个规则均要附加一个品质函数，用以评估规则的置信概率。

例如，自然语言"顾客进入商店要购买糖。"的形式化描述为"TRUE->购买糖"是一条规则，其置信概率可以由全部顾客进入商店要购买糖的概率（为3%）来表示。而对此规则进行扩充，又可以得到如下规则"购买咖啡的顾客->购买糖"（置信概率为16%）。

数据开采客户负责生成初始的一些规则，然后对其进行扩充。由于可以生成大量的规则，因此在实际应用中，对扩充的方法要进行限制，以减少所生成规则的数量，也就减小了问题搜索空间。

扩充后规则相对于其扩充前规则的品质函数值会有一定改变，出于资源的限制，我们只能对这一知识搜索树进行必要的剪枝，保留那些品质函数值改变较大的扩充后规则，称它们为新规则。在此基础上，由数据开采客户继续生成条件更详实的新规则，重复前述步骤，迭代到一定阈值的深度时，即终止，将所得到的规则转换为自然语言，即为知识。

### 2 基本的数据库操作

为了计算这些生成的规则的品质函数值，数据开采客户必须先将生成的规则解析成一些基本的数据库操作与一些比较、计算操作，然后将其中的数据库操作交付于后台的数据处理服务器来处理。

为了便于后台的数据处理服务器并行处理，我们将整个关系数据库中的关系模式分解成若干个仅包含关系模式中单个属性的二元关系表（简称为单域表 SAT）。一个单域表包含两列，第一列是元组（tuple）标识 tid，即对应数据库里元组的物理标号，第二列是该单域表所表示的属性。例如顾客信息数据库 C(CNum,Date,Time,Total)，它被分解为四个单域表 C.CNum (CTid, CNum)，C.Date (CTid, Date)，C.Time(CTid,Time)，C.Total(CTid,Total)。而出售商品数据库 G(GNum,Name,Type,Amount, Price, CNum) 被分解为六个单域表 G.GNum (GTid, GNum)，G.Name (GTid, Name)，G.Type (GTid,Type)，G.Amount(GTid,Amount)，G.Price (GTid,Price)，G.CNum(GTid,CNum)。由同一数据库分解出来的诸单域表彼此间通过 tid 联系，例如，单域表 C.CNum 有元组(00032964,000024)，单域表 C.Time 有元组(00032964,9:27am)，这两个元组的 CTid 相同，表明000024,9:27am 分别是顾客信息数据库里同一个元组的两个属性 Num 与 Time 的属性值。

我们定义了以下几个主要的基于单域表的数据库操作：

(1)COUNT

$$SAT.count=\sum_{tuple\in SAT}1$$

COUNT 是统计单域表中元组的数目。

(2)SELECT(bottom,top)

SAT.select(bottom,top)={(tid,value)|bottom≤value≤to;(tid,value)∈SAT}

SELECT 是从单域表 SAT 中选取那些属性值在给定的 bottom 与 top 之间的那些元组，生成新的单域表。当 bottom 等于 top 时，SELECT 是从单域表中选取那些属性值等于 top 的那些元组，生成新的单域表。

(3)UNION(SAT1,SAT2)

union(SAT1,SAT2)={(tid,value)|(tid,value)∈SAT1 or(tid,value)∈SAT2}

UNION 即将两个具有相同元组类型的单域表归并，此操作用于数据归类。

(4)SEMIJOIN(SAT1,SAT2)

semijoin(SAT1,SAT2)={(tid1,valuel)|∃ (tid2,value2)∈SAT2∧tid1=tid2∧tid1,tid2∈domain of the same attribute} or {(tid1,value1)|∃ (tid2,value2)∈SAT2∧value1,value2∈domain of the same attribute∧valuel=value2}

SEMIJOIN 即半联结。若两个单域表的第一列名字

相同，则半联结的结果是将两个单域表中元组标识相同的那些SAT1中的元组收集在一个新的单域表中，若两个单域表的第二列名字相同，则半联结的结果是将两个单域表中属性值相同的那些SAT1中的元组收集在一新的单域表中。

## 四、数据处理服务器

我们在数据处理服务器上建立了一个任务调度服务器，它负责接收来自数据开采客户发送来的数据库处理指令，然后按步就班地并行执行，最后将得到的结果回送给数据开采客户。

由于大量的单域表操作的可并行性很强，这就给并行计算机以很大的发挥余地。合理的调度策略是提高并行效率的关键。数据处理服务器上的任务调度服务器负责协调各处理机的任务分担。

在所有的数据库操作指令中，只有COUNT指令反馈结果，其它指令仅是生成中间结果。

下面我们研究顾客购买罐装液体类饮品与购买冲剂类饮品的相互影响。

数据处理服务器需要执行如下的数据库操作：

(1)C.CNum. Count 统计所有顾客的数目。

(2)tmp1＝G.Name. Select(Coca-Cola，Coca-Cola)可口可乐的商品元组表。

(3)tmp 2＝G.Name. Select(Tsingdao beer，Tsingdao beer)青岛啤酒的商品元组表。

(4)tmp3＝G.Name. Select(milk，milk)牛奶的商品元组表。

……其它罐装液体类饮品的商品元组表。

(m)tmpm＝Union(tmp1，tmp2)可口可乐或青岛啤酒的商品元组表。

……逐一合并罐装液体类饮品的商品元组表。

(n)tmpn＝Union(tmpw，tmpx)罐装液体类饮品的商品元组表，其中 w＝m－1，x＝n－1。

……诸冲剂类饮品的商品元组表。

……逐一合并冲剂类饮品的商品元组表。

(l)tmp1＝Union(tmpy，tmpz)冲剂类饮品的商品元组表。其中 n＜y，z＜1。

(o)tmpo＝Semijoin(G_CNum，tmpn)购买罐装液体类饮品的顾客元组表。

(p)tmpp＝Semijoin(G_CNum，tmp1)购买冲剂类饮品的顾客元组表。

(q)tmpq＝Semijoin(tmpo，tmpp)即购买罐装液体类饮品且购买冲剂类饮品的顾客元组表。

(r)tmpr＝Semijoin(C_CNum，tmpo)购买罐装液体类饮品的顾客元组表。

(s)tmpr. Count 购买罐装液体类饮品的顾客人数。

(t)tmpt＝Semijoin(C.CNum，tmpp)购买冲剂类饮品的顾客元组表。

(u)tmpt. Count 购买冲剂类饮品的顾客人数。

(v)tmpv＝Semijoin(C.CNum，tmpq)即购买罐装液体类饮品且购买冲剂类饮品的顾客元组表。

(w)tmpv. Count 即购买罐装液体类饮品且购买冲剂类饮品的顾客人数。

诸数据库操作的可并行性是显然的：(1)-(m-1)操作可以并行；(m)-(n)操作可以二分法，成对归并；(n＋1)-(l)同前；操作 o，p 可并行；操作 r，s 与操作 t，u 以及操作 v，w 又可以分别加载在三个处理器上。

上述四个数值返回给数据开采客户，数据开采客户进行如下计算：

购买罐装液体类饮品的概率＝tmpr. Count/C. CNum. Count

购买冲剂类饮品的概率＝tmpt. Count/C. CNum. Count

购买罐装液体类饮品后又购买冲剂类饮品的概率＝tmpv. Count/tmpr. Count

购买冲剂类饮品后又购买罐装液体类饮品的概率＝tmpv. Count/tmpt. Count

可以得到：

购买罐装液体类饮品的概率约为29%；

购买冲剂类饮品的概率约为11%；

购买罐装液体类饮品后又购买冲剂类饮品的概率约为2%；

购买冲剂类饮品后又购买罐装液体类饮品的概率约为5%；

即购买罐装液体类饮品后又购买冲剂类饮品的概率小于正常购买冲剂类饮品的概率；购买冲剂类饮品后又购买罐装液体类饮品的概率也小于正常购买罐装液体类饮品的概率，也就得出顾客对这两类饮品的选购是相互排斥的。商家在陈列饮品类商品时就应考虑这个因素。

## 五、实验结果

我们在国产全对称紧耦合共享存储、多处理机的曙光一号对一星期的近四万条顾客记录进行了处理。由于在数据装入内存前，数据已经被分解成若干单域表，因而每个处理机都能够处理大量的数据，并能将处理得到的部分数据保存在内存，其余的部分写到外存上，从而相应减少了单处理机不可避免的频繁的读写操作，提高了机器使用效率。

我们针对不同的参与数据库处理的处理机数目作了实验：

表 并行处理实验结果

| 处理机数目 | 处理时间(s) | 加速比 |
|---|---|---|
| 1 | 486 | 1 |
| 2 | 253 | 1.85 |
| 3 | 176 | 2.66 |
| 4 | 133 | 3.52 |

可以观察到，单处理机在处理大规模数据库时，由于资源上的限制，不得不频繁地将数据从内存卸载到外存上，再从外存上载入内存，降低了处理效率。即便给单处理机配置大容量的内存，但由于其内存的使用效率很低，因而极不经济。目前，国际上并行机技术发展迅速，巨型并行处理机的处理能力远远超过理论上的单处理机的极限能力。而且从资源上的使用效率而言，并行机也胜于单处理机。我们实验用的国产曙光一号有四个 CPU，共享64MB 内存，CPU 采用的是摩托罗拉公司的 RISC MC88100。

**结束语** 基于客户/服务器结构的数据开采对于大型数据库的处理确实具有其优越性，由于客户端与服务器端分工明确，能够各自发挥所长。客户端着重开发可视化的用户界面与规则生成分析部分；服务器端着重开发数据并行处理部分。而相对于整个从数据库中发现知识过程，数据的前处理与后处理也可以集中在客户端来处理。特别是在当今世界并行机的飞速发展，已经不再是单处理机所能企及，因而基于客户/服务器结构的数据开采技术也就具有非常重要的实际应用意义。

## 参考文献


[1] Fayyad U. M. et al., Knowledge Discovery and Data Mining: Towards a Unifying Framework. Proc. 2nd Intl. Conf. on Knowledge Discovery and Data Mining (KDD-96). Menlo Park, CA: AAAI Press. 1996

[2] Piatetsky-Shapiro G. et al., An Overview of Issues in Developing Industrial Data Mining and Knowledge Discovery Applications. Same to [1]

[3] Fayyad U. M. et al., From Data Mining to Knowledge Discovery: An Overview. in AKDDM. AAAI/MIT Press. 1996

[4] Holsheimer M. et al., Data Surveyor: Searching the Nuggets in Parallel. Same to [3]


（上接第22页）

——空 Herbrand 解释满足→p——于是第二条规则派生出 q。空解释的含义是计算程序 P 语义的初始点，正常地它应表示我们对 P 定义的正和负事实一无所知，于是这成为坚持模型论语义的第二点理由。

虽然人们针对逻辑程序的语义提出了多种的方法，但似乎总可以找到实例使已知方法不能抓住程序的"自然"的主观含义，于是人们一边定义了各种满足某种性质的程序类，一边无休止地继续"繁殖"语义方法，虽然人们对各种语义模型的关系做了不少的研究，对语义模型的统一框架的研究也有完整的结果，但主要的说明语义如何"结合"和"转换"是一个实际的问题，这种"结合体"也许会有更强的反映常识和主观含义的能力。

表5.1 静止语义统一了各种模型论语义[1]

| 语义名称 | not C 的翻译 | 附加公理 |
|---|---|---|
| 良基 | B→C | |
| 静态 | B→C | |
| 稳定 | B→C | (BCA) |
| 稳定 | →LC | |
| 静止 | B→C | |
| 扩展静态 | B→C | (S) |
| 扩展静态 | B→C | (S),(DBA) |
| 扩展稳定 | →LC | (S) |
| 扩展静止 | B→C | (S),(DBA) |
| 折取静态 | B→C | (S),(DBA) |
| 折取稳定 | →LC | (S),(PIA) |


## 参考文献

[1] Van Gelder. A. et al. The well-founded semantics for general logic programs. J. ACM 38(3). 1991

[2] 王怀民，逻辑程序的语义问题(I)(II)，计算机科学，21(1)，21(2)，1994

[3] Dung. P. M., On the relation between stable and well founded semantics of logic programs. Theoretical Computer Science. 105(1). 1992

[4] Przymusinski. T. C., Three-valued non-monotonic formalisms and semantics of logic programs. Artificial intelligence 49. 1991

[5] Marek. W. and Subrahmanian. V. S., The relationship between stable, supported, default and autoepistemic semantics for general logic programs. Theoretical Computer Science. 103. 1992

[6] Gelfond. M., Lifschitz. V., Classical negation in logic programs and disjunctive databases. New generation computing. 1991

[7] Przymusinski. T., Extended stable semantics for normal and disjunctive programs. Proc. of 1990 intl. conf. on logic programming. 1990

[8] Przymusinski. T., Stationary semantics for disjunctive logic programs and deductive database. Proc. of the 1990 American conf. on logic progcamming. 1990

[9] 陈莽，孙吉贵，不相容逻辑程序的矛盾处理及其语义 AFSX(待发表)

[10] Przymusinski. T., Semantics of normal disjunctive logic programs--a unifying framework. ICLP' 94 workshop. 1994


(1)语义名称前可能增加了"折取"或"扩展"的字样，这对应于折取或扩展逻辑程序；否则，表示该语义是常规逻辑程序的语义。