SNMP 数据采集引擎的采集算法及其优化*)

赵成栋 周 宇 康建初 谭 俊
(北京航空航天大学计算机系 北京100083)

Collect Algorithm and Optimization for SNMP-Based Data Collect Engine

ZHAO Cheng-Dong ZHOU Yu KANG Jian-Chu TAN Jun
(Computer Science & Engineering Dept, Beijing University of Aeronautics and Astronautics, Beijing 100083)

Abstract SNMP-based data collecting is fundamental for IP network management. This paper presents a SNMP-based Data Collect Algorithm for common use. According to requirement of data collecting, the algorithm disassembles collect tasks into metadata, and then reconstructs the protocol packet. Also takes full advantage of GetBulk operation, and brings out a combination algorithm for discrete variables of a MIB table so that the operation is optimized. A collect engine is implemented using the algorithm, which supports dynamic collection of multiple MIB variables, supports arithmetic computing and logical computing, supports threshold judgment and sends out abnormal events notifications. Tests are given to prove that the algorithm reduces the traffic for network management and improves the efficiency.

Keywords Network management, Data collect, SNMP, MIB, GetBulk, Polling

1 引言

基于 SNMP 的数据采集引擎是网络管理系统的重要组成部分，是关系到网络管理准确性和有效性的关键。为了不断地从被管理网元中获取实时的动态变化的管理数据，引擎通常以简单网络管理协议 SNMP(Simple Network Management Protocol)作为其数据采集支撑协议[1~3]，以轮询(Polling)的方式反复地执行数据提取操作，获取包括性能、故障、配置、计费和安全等数据在内的各类数据，并将其提供给其他网络管理系统，进行进一步的分析处理，以便能最终向网络管理人员报告网络的实际运行状态。

基于 SNMP 协议进行数据采集的研究[4~7]和实现方面，许多大公司，如 HP 公司的 NNM:Network Node Manager，已做了大量的工作。在使用新的数据获取方法方面，Michael 等提出了使用移动代理(Mobile Agent)的数据采集法[4~5]，但由于移动代理需要运行环境支持，如 Java 虚拟机上的 Aglet 或其他的一些实现，目前情况下，这种方法的应用还受到很大的限制。其他如基于 HTTP、WebService 的方法[6]，由于缺乏网络设备环境(Agent)的支持，目前也不能广泛采用。Hwang 提出了基于组播的 SNMP 采集方法[7]，虽然在一定程度上可减少请求报文的数目，但是由于组播方式带来的子网流量泛洪(flood)，可能导致被管理网络性能的急遽恶化[11,12]，实践中也不是一个有效的应用方法。

目前，能够在工业环境中得到广泛应用的仍然是基于传统 Manager-Agent 模型[8]及其改进模型。然而，即便是基于 M-A 模型的研究和实现方法，还存在大量待改进和优化的余地。例如在系统结构上，由于每个网管应用进程均通过 SNMP 协议直接访问设备代理而获取数据。这种分散的访问中存在着对大量同一变量的重复读取，不仅操作效率较低，而且加大了设备代理的负载和对网络带宽的消耗，特别是在同时存在多个网管应用和采集频率高的情况下，可能会恶化被管理网络的性能；在低带宽的网络环境中，可能导致网络的崩溃[10]。在数据采集应用功能上存在以下问题：

① 不支持 MIB 变量动态组合，灵活性差。很多应用实现使用特定的独立进程对一个或者多个特定 MIB 变量进行采集，不仅系统难以扩展，且复用性低。

② 不支持 MIB 变量之间的四则算术运算和逻辑运算。在进行 SNMP 数据采集时，一些基本 MIB 变量的值经过一定运算后得到的结果对提供管理信息会更有用，如果不加运算就传输数据，不仅会造成大量原始数据(raw data)的存储和传递，而且会造成更大的延时，降低了数据的时间有效性。

③ 不支持基于阈值的告警(Threshold Alarm)，或者支持的能力弱。SNMP 的数据采集对象很大一部分是性能参数，在性能参数超出正常取值区间时，表明网络性能恶化，需要及时报告给网络管理人员或者其他网管应用系统，因此 SNMP 采集系统对阈值的支持非常必要。

本文提出了一个基于 M-A 模型的 SNMP 数据采集算法，并将其应用于一个采集引擎，使此引擎支持 SNMP v1，v2c 所有类型的 MIB 变量的采集，支持 MIB 变量之间的四则算术运算和逻辑运算；当运算结果满足阈值告警条件时，可以实时地发送用户自定义的告警信息；并可将需要保存的数据自动保存到数据库中。为了提高数据采集的效率，降低网络负载，本文还对采集算法进行了优化处理。

文中介绍了引擎的总体设计、数据采集算法及其优化，并给出了实验对比结果。

2 基于 SNMP 的数据采集引擎

基于 SNMP 的数据采集引擎是一个能够自动从网络环境中获取 SNMP MIB 变量值的独立运行系统，其结构如图1所示。其他的模块或系统可以通过通信接口将自己对 SNMP

*)本课题得到国家九七三项目("网络环境下海量信息组织与处理的理论与方法研究"中的"基于先进网络的大型电信网络管理示范系统"子项目)的资助(G1999032709)。

MIB 变量的采集需求作为采集任务发送给采集引擎，采集引擎通过 SNMP 协议从被管理实体(物理设备、软件系统)中获取相应数据，并将其保存到数据库中或者返回给采集任务的请求者。

在数据采集的过程中，任务的请求者可以对任务的参数进行调整，或者停止对数据的采集。请求者还可通过发送控制命令的方式，动态地改变采集引擎的采集任务的配置参数。

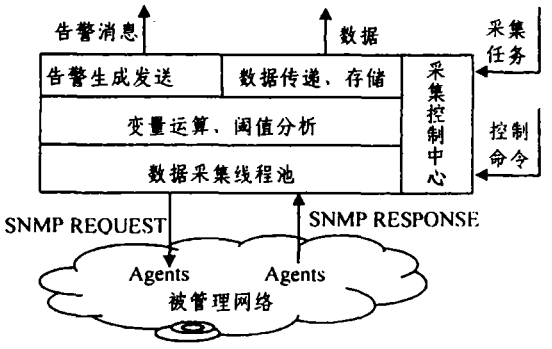

图1 SNMP 数据采集引擎系统框架

3 引擎的数据采集算法

3.1 概述

引擎的数据采集算法执行过程如图2所示。

一般来说，一个网络管理应用在网管对象的属性、操作等与 SNMP MIB 变量之间建立映射，这里的 MIB 变量可以分为单元变量、变量算术表达式和逻辑表达式三种，并把相应的网络设备操作和参数值获取等处理作为一个 SNMP 数据采集任务通知数据采集引擎。其中，一个实时采集需求(On-Demand)可被视为一个频率为0的任务。

由于同一网元设备发出的多个任务中的对象标识符 OID (Object IDentifier)可能会有大量的重合，如果每个任务单独发送报文进行设备数据采集，不仅需要重复使用 OID 的数据，而且会由于包含 OID 较少的一般性任务的大量出现，而产生许多的短小报文，占用更多的网络带宽。如若将同一网元设备发出的多个任务中的对象标识符 OID 组装在一个请求报文中，可以有效地减少重复访问，从而降低网络管理造成的网络流量(以下简称网络流量)。在对这些 OID 归并过程中，如果遇到对 MIB 表格的采集，还可以使用 GetBulk 操作，将原来需要发送多个 GetNext 的操作简化成对一个报文的操作，以此极大地降低网络流量。引擎的数据采集算法充分利用了 GetBulk 操作的优势，降低网络流量，提高数据采集效率。

3.2 SNMP GetBulk 原语

众所周知，SNMP v1定义了 Get、GetNext 两种用于请求数据的操作原语。系统在进行 MIB 表格数据采集时，需要不断地发送 GetNext 请求，才能完成一个表格的数据采集过程，其效率很低[13,14]。

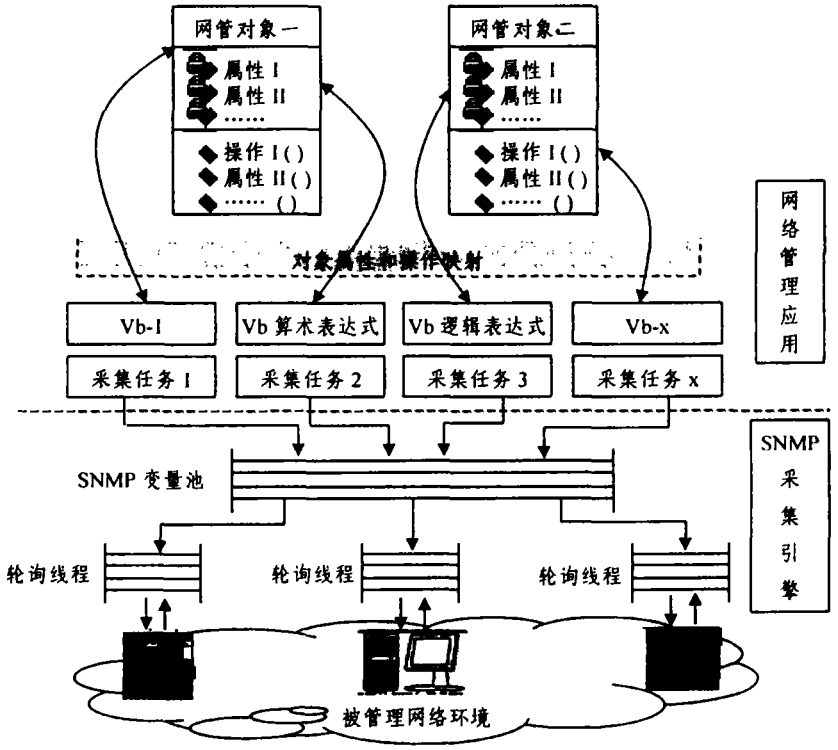

图2 数据采集算法示意

SNMP v2为了提高对 MIB 表格的操作效率，引入了 GetBulk 操作原语[2]，原语的 PDU 格式如图3所示。该原语可用于连续获取 Variable-bindings 中多个变量的 max-repeations 个后继变量的值，从而可大大地提高协议操作的效率，降低网络带宽的使用。

3.3 术语定义

为了方便后面算法的描述，首先引入以下几个定义。

定义1 一个 SNMP 数据采集任务可以看作一个五元组：$Task=(Agent, f, OidSet, FormulaSet, LogicalExpSet)$ 其中，*Agent* 是任务要采集的对应设备代理的信息(IP 地址、Community 等)，f 是任务执行的频率，*OidSet* 是任务要采集的 MIB 变量的集合，*FormulaSet* 是任务针对采集变量结果进行计算的公式集合，*LogicalExpSet* 是任务进行逻辑判断的表达式集合。

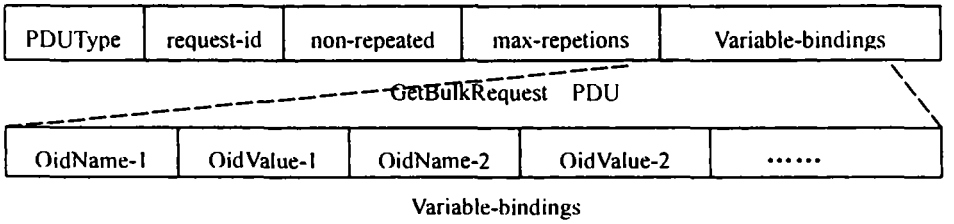

图3 SNMP GetBulk 报文格式

定义2 *OidSet* 是由被采集 *OID* 变量及其相关信息项组成的集合：

$OidSeti=\{OidTuple \mid OidTuple$ 是 $Task_i$ 要采集的 *MIB* 变量组成的元组$\}$

下标 i 区分不同的任务及其对应的子项(下同)。

定义3 *OidTuple* 是包含一个 *OID* 变量及其相关信息项的元组：

$OidTuple=(OidStr, OidStored, OidStoreTable, OidStoreField, OidFieldType, OidAlerted, OidThresholdCmpExp, OidAlertType, OidMsgMacro)$

其中，*OidStr* 是被采集 MIB 变量 *OID* 的字符串表示，*OidStored* 指示对应的 MIB 变量值是否已存储到数据库中，*OidStoreTable* 是该 MIB 变量值存储的数据库中的表名称，*OidStoreField* 是该 MIB 变量值存储的数据库中的字段名称，*OidFieldType* 是该 MIB 变量值存储的数据库中的数据类型，*OidAlerted* 指该 MIB 变量是否进行阈值判断和告警，*OidThresholdCmpExp* 是进行阈值判断的表达式，如：$Min \leqslant Value < Max$。这里的 *Min*、*Max* 组成的区间给出了变量变化的有效范围，变量值一旦超出此范围，系统将会触发预定义的告警消息。*OidAlertType* 是发送阈值告警信息的告警类型，*OidMsgMacro* 是包含阈值告警信息的宏，可以在发送告警信息时进行宏替换，成为可读信息，发送给其他网管应用。

类似地，还可以给出 *FormulaSet*、*LogicalExpSet* 的定义(本文从略)，它们也是建立在 *Oid* 采集结果的基础上的包含存储标识和告警标识的元组集合，以便进行进一步的计算和判断。

定义4 进行数据采集的基本 MIB 变量元组 *VbTuple* 是这样一个元组：$VbTuple=(Agent, f, OidStr)$。

其中，*Agent* 是被采集设备代理的信息，f 是进行采集的频率，*OidStr* 是被采集 MIB 变量 *OID* 的字符串表示。此元组是进行 SNMP 数据采集的原子信息。

3.4 数据采集算法

如图3所示，SNMP 采集引擎在进行数据采集之前，需首先将所有采集任务进行分解，把其所包含的变量分离出来，组成一个 MIB 变量池；然后按照 *Agent* 和频率 f 的不同，将其加入不同的数据采集队列，重新组装 SNMP 报文；再启动线程发送和接收 SNMP 报文，获取所需的数据；当数据返回之后，算法按照原定任务的设置，对数据进行计算、阈值分析、发送告警、存储等处理。

算法具体的步骤如下：

1)初始化，获取所有的任务集合 *TaskSet*。

$TaskSet=\{Task_i \mid Task_i$ 是数据采集引擎承担的采集任务$\}$

2)从 *TaskSet* 的每一个 *Task* 中分解出 *VbTuple* 元组，组成集合 *VbSet*。

$VbSet_i=\{VbTuple_{ij} \mid VbTuple_{ij}$ 是从 $Task_i$ 中分解出的 *VbTuple* 元组$\}$

$VbTuple_{ij}=(VbAgent_{ij}, Vbf_{ij}, VbOidStr_{ij})$

其中 $VbAgent_{ij}=OidTupleij.Agent_{ij}$，$Vbf_{ij}=OidTupleij.f$，$VbOidStr_{ij}=OidTupleij.OidStr$。

3)建立 *Vb* 变量池 *VbPool*。

$VbPool=\{VbTuple_{ij} \mid VbTuple_{ij} \in VbSet_i, i=1,2,\cdots N.$ N 为采集任务的个数 $\}$

4)按照代理信息 *Agent*、采集频率 f 进行分类，建立不同的采集队列集合 $VbQueue_1 \cdots VbQueue_m \in VbPool/R$，$VbPool/R=\{[VbTuple]_R \mid VbTuple \in VbPool\}$

$R=\{\langle VbTuple_x, VbTuple_y \mid VbTuple_x.Agent=VbTupley.Agent \wedge VbTuplex.f=VbTupley.f\rangle\}$

5)对每个数据采集队列 *VbQueue*，建立集合 *OidPackage*

$OidPackage_i=\{OidStr_{ij} \mid OidStr_{ij}$ 是 $VbQueue_i$ 中 $VbTuple_{ij}$ 的 $OidStr_{ij}\}$

并为该 *VbQueue* 构造 SNMP 请求报文，其中 $OidSt_{rj} \in OidPackage_i$：

5.1)如果 $OidStr_{ij}$ 是一个 MIB 表格的根节点，表明是要获得该表格的所有变量数据，构造一个 *GetBulk* 报文，取该 $OidStr_{ij}$ 作为报文绑定的参数项。

5.2)对其他的 *OidStr* 变量组合成一个或者多个(为了防止报文过大溢出)SNMP *Get* 报文。

6)为每个 *VbQueue* 建立定时机制。达到规定时间时，启动一个线程，进行 SNMP 数据采集。

7)基本数据采集完成后，启动算术表达式、逻辑表达式和阈值处理模块，按照表达式要求进行组装、计算、判断。

8)对计算结果进行阈值分析、发送告警，并将结果保存到数据库指定的位置中。

3.5 离散表格的数据归并优化算法

引擎在进行 SNMP 数据采集过程中，将任务分解后，获得的单个采集变量队列可能会出现如图4(A)所示的情况(图中，表格表示 MIB 中定义的变量关系，其中列代表不同的变量，行代表变量的不同实例值，阴影部分表示需要采集的 MIB 变量值)，即被采集的一组变量虽在同一个 MIB 表格中，但它们是离散的、不连续的。

上述算法步骤5.2 由于忽略了同一个表格变量之间存在着这些内在关系，故其数据采集效率仍然不高。本文进一步提出了离散表格数据的归并采集算法，将对每个变量的离散实例的分别提取视为对一个表格进行的数据提取，利用 GetBulk 操作获取数据，很好地解决了上述问题，进一步提高了采集的效率。

定义5 函数 INSTANCE_OF_OID(OID_STR)是获取一个 MIB OID 的实例编号，如 OID 1.3.6.1.2.1.2.2.1.10 是 MIB II 中叶节点 ifInOcets 的 OID，它的第一个实例的

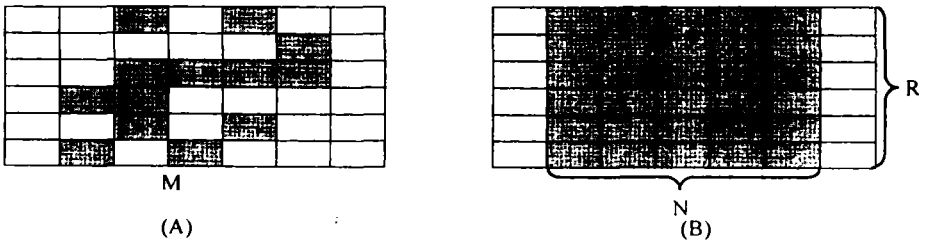

图4 离散表格变量组成 Bulk 表格变量示意图

OID 为1.3.6.1.2.1.2.2.1.10.1,则

INSTANCE_OF_OID (" 1.3.6.1.2.1.2.2.1.10.1 ") =1

定义6 函数 TABLE_COLOMN_OF_OID(OID_STR) 是获取一个 MIB OID 的在 MIB 表格中对应的列序号,如 OID 1.3.6.1.2.1.2.2.1.10是 MIB II 中叶节点 ifSpeed 的 OID,则

TABLE_COLOMN_OF_OID ("1.3.6.1.2.1.2.2.1.10") =10

归并补充算法:

1)根据 MIB 定义判断将 OidStr 进行归类,在同一表格中的变量组成一个表格变量集合。不能进行表格归类的,组成一个非表格变量集合。

2)对于每一个表格变量集合,如 OID_SET_i,设共有 M 个被采集 MIB 变量,跨越某 MIB 表格的 R 行、N 列(如图4(B))。N、R 的值使用下面的表达式确定:

$$N = Max\{j \mid j = TABLE\_COLOMN\_OF\_OID\ (oid), oid \in OID\_SETi\}) - Min\{\ j \mid j = TABLE\_COLOMN\_OF\_OID\ (oid), oid \in OID\_SETi\} + 1$$

$$R = Max\{j \mid j = INSTANCE\_OF\_OID\ (oid), oid \in OID\_SETi\}) - Min\{j \mid j = INSTANCE\_OF\_OID\ (oid), oid \in OID\_SET_i\} + 1$$

3)对每个表格变量集合组装 SNMP 报文,使用 GetBulk 进行提取。将 GetBulk 的报文字段 non-repeated 置0,字段 max-repetitions 置为 R,将 N 个被提取的互不相同的变量填入到报文变量区域。

4)非表格变量集合仍然采用 SNMP *Get* 进行提取。

可见,采用归并补充算法替换3.2节中算法的步骤5.2,可以得到数据归并采集算法,实现数据采集算法的优化。

下面对优化前后的两个算法的操作效率进行比较。

a)使用非优化算法:将 M 个变量合并到一个 SNMP *Get* PDU 报文中(假定 M 不超过 SNMP PDU 的最大长度),在发送 REQUEST PDU 中填充 M 个 VariableBind(以下简称 VB),并接收含有 M 个 VB 的 RESPONSE PDU 报文。期间,报文流量(为简化起见,不讨论 PDU 报文头部产生的流量,只讨论 VB 的数量)为:

$$M + M = 2M$$

b)采用归并优化算法:将该 M 个变量的采集归并成 N * R 的表格操作,使用 GetBulk 进行提取。GetBulk 的报文填充 N 个 VB,发送后,会得到含有 N * R 个 VB 的 RESPONSE PDU。期间,报文的流量为:

$$N + N * R = (R+1) * N$$

从N、R 的定义,可以得到:

$$M \leqslant N * R$$

令

$$2M \geqslant (R+1) * N$$

可得:

$$R \geqslant 1$$

可见,只要 M 个变量中包含不同表格行的 MIB 变量,优化后引发的流量就低于不优化的流量。显然,这个条件非常容易满足,如一般设备的 ifEntry 都包含多行,进行管理时就会对多个接口的属性进行数据采集。

3.6 算法的测试结果

本文在 Windows2000服务器(PIII800/256M 内存/100M 网卡)上,利用数据引擎的数据归并采集算法和传统的网管应用系统通常使用的数据采集算法,针对不同的被采集 MIB 变量数目发生的网络流量进行了测试和相关的性能对比。测试结果如图5所示,其中横轴表示 MIB 变量的数目,纵轴是采集时发生的流量。可以看到,本文提出的采集引擎产生的网络流量大大低于传统算法。

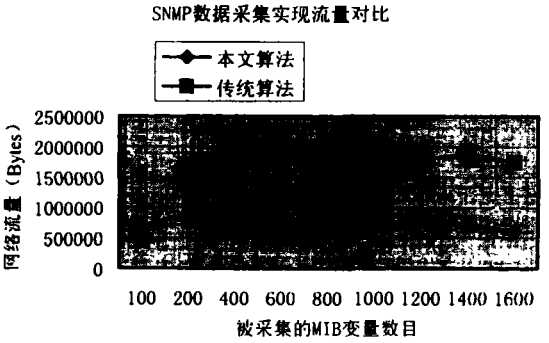

图5 试验结果分析图

结论 为了改善 SNMP 数据采集的效率和提供更有效的功能,本文提出了一个通用的 SNMP 数据采集算法。该数据采集算法充分利用了 SNMPv2提供的 GetBulk 操作的优势,并提出了 MIB 离散表格数据的归并算法,从而获得了优化。实验证明,使用该引擎进行数据采集,降低了网络管理产生的流量,提高了采集效率。

参 考 文 献

1 McCloghrie K, Rose M T. Management Information Base for Network Management of TCP/IP-based internets:MIB II. RFC1213,1991

2 Rose M T. Mangement Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2). RFC1907, 1992

3 Harrington D, Presuhn R, Wijnen B. An Architecture for Describing SNMP Management Frameworks. RFC2271,1998

4 Zapf M, Herrmann K, Geihs K, Wolfgang J. Decentralized SNMP management with mobile agents. In Sloman et al.

点击流中事务数据模型的设计与实现*)

辛 燕 鞠时光 蔡 涛 阎星娥
(江苏大学 镇江212013)

Design and Implement of Transaction Data Model in Clickstream

XIN Yan JU Shi-Guang CAI Tao YAN Xing-E
(Jiangsu University, Zhenjiang 212013)

Abstract In this paper, we first briefly introduce the concepts of clickstream data and data warehouse, analyze two existing clickstream star schema——click star schema and session star schema in webhouse, then induce a new model——transaction star model based on them, and expressed the method of bringing out the model. Comparing with the two schemas mentioned above, its most apparent speciality is that it includes a series of meaningful page-view sequence rather than a single click. Thus, on the one hand it improves the query performance of data, on the other hand it is in favor of executing more deepen analysis——data mining, and simplifies the process of data pretreatment. At last, the paper verifies its' feasibility and validity using association rules based on the model.

Keywords Clickstreams, Data warehouse, Star schema, Transaction fact model, Association rules

点击流数据简单说就是Web服务器上一系列有序的日志记录。随着WWW应用及电子商务的高速发展，电子商务网站的Web服务器上自动收集了大量的用户访问信息记录，即所谓的Web日志。Web日志蕴涵了大量的有用信息，如客户来源、客户访问趋势、客户兴趣、网站流量等，因而记录和分析Web日志数据已逐渐成为e企业的一项重大活动。点击流数据仓库对原始的Web日志数据进行过滤、清洗并集成，以便于利用联机分析处理和数据挖掘技术对点击流数据做进一步分析，从而为企业创造巨大的信息财富。

本文主要针对点击流数据分析中的一次事务访问过程，提出对点击流数据进行事务事实建模，即为单个的事务访问建立数据分析模型。文章在分析Kimball提出的两种星型模式——点击星型模式和会话星型模式的基础上[1]，设计了事务星型模式，并给出了实现方法，最后通过关联规则挖掘在该模型上的应用来说明其在点击流数据分析中的可行性及有效性。

1 点击流数据与点击流数据仓库

顾客从进入一个电子商务站点，到离开这个站点的一个访问周期中，所浏览的页面、滞留时间、点击的链接和广告都会被顺序地记录在网站的日志文件中。这种有序的Web日志记录形成了所谓的点击流数据。而在Web服务器日志文件中，无论是普通日志格式还是扩展日志格式，都包括客户端IP、用户名/用户ID、请求时间、请求方式、请求的目标文件名、状态、参引页、代理、cookie等基本的数据域。但是在点击流数据分析中，并非每一个数据域都是有用的数据域，如请求方式、状态等数据域对分析过程并没有多大作用，因而具体分析时可将无关的数据域剔除掉。

为了对点击流数据进行分析，通常需为原始点击流数据建立数据仓库。这是因为：①点击流数据来源于Web服务器上原始的日志记录，对于一个较大的电子商务网站来说，每天将产生数以百万条计的日志记录，数据仓库本质上来说就是一种大型的数据库，因而能有效地组织和管理大量的点击流数据；②原始的点击流数据中含有一些无用的数据成分，因此在分析点击流数据之前，需要对其进行适当的预处理，数据仓库中的数据一般是经过抽取、转换、清洗与集成了的合成数据，因而在数据仓库上进行点击流数据分析可免去好多数据预处理的工作；③数据仓库中集成了大量的历史数据，而点击流数据分析中好多也都与时间有关，如点击序列模式分析等。因而，建立面向点击流的数据仓库，已成为点击流数据分析中

*)本课题的研究得到国家自然科学基金(60173064)及江苏省自然科学基金(NO. BK200204)的资助。

5 Pagurek B, Wang U, White T. Integration of mobile angents with SNMP: Why and how. Submiited to NOMS'2000, 2000
6 Martin-Flatin J-P, Bovert L, Hubaux J-P. JAMAP: a web-based management platform for IP networks. In: Proc. of the 10th IFIP/IEEE Intl. Workshop on Distributed System: Operations and Management(DSOM'99)
7 Hwang K-C, Hong J-J, Lee, K-H. A SNMP Group Polling for the Management Traffic. TENCON 99. In: Proc. of the IEEE Region 10 Conf. 1999, 2: 797～800
8 Case J, Fedor M, Schoffstall M, Davin J. A Simple Network Management Protocol (SNMP). RFC1157, 1990
9 Stallings W. SNMP, SNMPv2, SNMPv3, and RMON 1 and 2(3rd edition). Addison-Wesley Pub Co, 1998
10 Chelkhrouhou M, Labetoulle J. An Efficient Polling Layer for SNMP. Network Operations and Management Symposium, 2000. NOMS 2000. 2000 IEEE/IFIP, 2000. 477～490
11 Jacquenet C, Proust C. An introduction to IP multicast traffic engineering Universal Multiservice Networks, 2002. EDUMN 2002. 2nd European Conference on, 2002. 263～267
12 CISCO. CGMP Isn't Preventing the Flooding of Multicast Packets for Certain Group Addresses. http://www.cisco.com/warp/public/105/mcastguide9.html
13 白彩英,等. 计算机网络管理系统设计与应用. 北京：清华大学出版社，1998
14 Sprenkels R, Martin-Flatin J-P. Bulk transfer of MIB data. The Simple Times, The Quarterly Newsletter of SNMP Technology, Comment, and Events, 1999, 7(1): 1～8