基于垂直数据分布的大型稠密数据库快速关联规则挖掘算法

崔 建 李 强 杨龙坡
(空军雷达学院预警监视情报系 武汉 430019)

摘 要 为进一步解决对大型事务数据库进行关联规则挖掘时产生的CPU时间开销大和I/O操作频繁的问题,给出了一种基于垂直数据分布的改进关联规则挖掘算法,称为VARMLDb算法。该算法首先有效地把数据库分为内存可以满足要求的若干划分,然后结合有向无环图和垂直数据形式diffset差集来存储和计算频繁项集,极大地减少了存储中间结果所需的内存大小,解决了传统垂直数据挖掘算法对稠密数据库挖掘效率低下的问题,使该算法可有效地适用于大型稠密数据库的关联规则挖掘。整个算法吸取CARMA算法的优势,只需扫描两次数据库便可完成挖掘过程。实验结果表明该算法是正确的,在大型稠密数据库中,VARMLDb算法具有较高的执行效率。

关键词 CARMA算法,DAG,diffset差集,垂直数据分布,稠密数据库
中图法分类号 TP311 **文献标识码** A

Fast Algorithm for Mining Association Rules Based on Vertically Distributed Data in Large Dense Databases

CUI Jian LI Qiang YANG Long-po
(Department of Early Warning Surveillance Intelligence, Air Force Radar Institute, Wuhan 430019, China)

Abstract To further reduce both CPU and I/O overhead in the process of mining the association rules on the large transaction database by the traditional algorithm, an improved algorithm of association rule mining based on vertical data layout named VARMLDb(Vertical Association Rule Mining for Large Databases) was suggested. In the proposed algorithm, after dividing the database into several partitions each of that is suitable for the current memory, the algorithm combines directed acyclic graphs and diffset(difference of tidlist sets) which belongs vertical data layout structure for storing and computing frequent item sets, which not only greatly cuts down the required memory size used to save intermediate results but also solves the low efficiency problem during the mining dense database by traditional vertical data mining algorithm, so that the algorithm is more effective for large dense databases. As a result of drawing the advantages of CARMA(continuous association rule mining) algorithm, the algorithm needs to scan the database for only twice. Experimental results show that the algorithm is correct, and in the large dense transaction databases, VARMLDb algorithm has higher implementation efficiency.

Keywords Continuous association rule mining algorithm, Directed acyclic graphs, Diffset plumb, Vertically distributed data, Dense databases

1 引言

关联规则是由Agrawal等人提出的数据挖掘的一个重要研究方向。Agrawal于1993年首先提出了挖掘顾客交易数据库中项集间的关联规则问题。近年来,随着商业和科学数据库急剧增长及存储设备不断升级,大型事务数据库的关联规则挖掘成为数据挖掘领域中一个非常重要的研究课题。

经典的数据库关联规则挖掘算法主要包括:Agrawal等提出的Apriori[1]算法、Park等提出的DHP[2]算法、Brin等提出的DIC[3]算法等。但这些算法在对大型数据库进行关联规则挖掘过程中,通常无法将驻留磁盘的数据一次性读入内存,造成大量的磁盘I/O操作和严重的CPU开销。之后有许多改进方法被提出,如AprioriTid[4],AprioriHybrid[4],Partition[5],Cloum-Wise Apriori[6],Multiple-Leve[7]等;国内学者丁艳辉针对大型交易事物数据库提出了一个新的关联规则挖掘算法BOM算法[8],该算法不需在内存中存储大量的候选项集,对于大型事务数据库具有较高的性能。

尽管基于Apriori算法改进的各种水平挖掘算法对于频繁模式长度较短的稀疏数据库有良好的表现,但是,对于具有较长频繁模式的稠密数据库,如人口普查数据库,这些算法的性能下降较为严重。下降的原因是这些算法在执行过程中多次扫描数据库,产生了大量的长频繁项集。这一方面导致多次扫描磁盘驻留数据库带来的较高I/O开销;另一方面,在对大候选项集集合进行模式匹配时,特别是挖掘长模式时,产生了较高的CPU运算开销。例如,一个长度为m的频繁模式暗含另外2^m-2个频繁模式需要计算,当m较大时,频繁项

到稿日期:2010-05-28 返修日期:2010-10-01 本文受国家自然科学基金项目(60736009)资助。
崔 建(1981—),男,博士生,工程师,主要研究方向为军事情报分析,E-mail:ldxy-cj@163.com;李 强(1968—),男,教授,博士生导师,主要研究方向为军事情报分析、并行计算、中间件;杨龙坡(1975—),男,博士生,讲师,主要研究方向为军事情报分析。

集挖掘算法的CPU开销将成为瓶颈。

为此，目前有多种垂直关联规则挖掘算法被提出，如VIPER[9]，MAFIA[10]，SPAM[11]算法等。国内也有部分学者对垂直方式的关联规则挖掘算法展开了研究，如宋长新和耿晓斐分别在文献[12,13]中对Eclat算法[14]进行了详细的讨论并对其进行改进，使改进后的算法对数据集较小的稠密数据库具有较好的挖掘性能。这些算法相比水平挖掘方法而言具有更高的执行效率，优点在于垂直挖掘算法利用tidlist[9]集合求交的方式替代传统方式中通过复杂的内部数据结构来计算频繁项集，对于大型事务数据库，垂直方法可以有效地减少I/O操作的数量。

尽管基于垂直方式的挖掘算法具有很多优点，但是当tidlist集合变得非常大时，这些方法的性能就会下降[15]。因为完成交集运算的时间将增多，同时产生大量的频繁项集的中间tidlist集合，造成内存不足，需要压缩数据或临时写入磁盘数据，所以针对稠密数据集，尤其是数据集具有普遍的长频繁模式时，传统垂直挖掘方法的性能下降较为严重，上述方法都未能很好地解决对大型稠密数据库的关联规则挖掘。

为解决以上问题，本文在前人工作的基础上提出了快速挖掘大型稠密事务数据库的高效算法VARMLDb(Vertical Association Rule Mining for Large Databases)。该算法结合Partition算法的优势改变了CARMA[16]算法逐个元组扫描的方式，按照内存大小对数据库进行分块读取，且每个划分只需扫描两次，大大减少了磁盘I/O操作；同时本文算法使用DAG数据结构存储频繁项集，并结合垂直数据形式diffset来降低存储中间结果所需的内存数量，利用diffset形式代替传统垂直算法中的tidlist形式存储初始数据库可以减少数据库的大小，使算法在稠密数据库中diffset的交集运算比tidlist的交集运算在性能上提升几个数量级，从而进一步提高了单个划分关联规则挖掘的效率，使本文算法在整体上能够在减少I/O操作和CPU时间开销方面都具有良好的表现。

实验表明，本文算法能够较好地适应大型稠密数据库中关联规则的挖掘。

本文第2节主要介绍相关概念与算法所涉及的数据结构；第3节详细介绍VARMLDb算法思想及其算法流程；第4节是实验结果和分析；最后是总结。

2 相关概念与描述

设项集集合为I，对给定事务数据库$T=\{t_1,\cdots,t_n\}$，记D为事务T的集合，对应每一个事务有唯一的标识TID，F为D的频繁项集集合，$P_1,\cdots,P_n$为D的n个互不相交的划分，d为算法执行过程中除当前划分之外的已扫描事务，$\mathcal{Q}$用以表示存储候选项集的DAG结构，v表示$\mathcal{Q}$中项集，即有$v\subseteq I$，定义v的支持度计数为$count(v)$，v的支持度为$support(v)$，$minsup$为用户设定的最小支持度门限。

2.1 频繁项集

定义1 给定事务数据集D和最小支持度$minsup$，对于项集$v\subseteq I$，若$support(v)\geqslant minsup$，则称$v$为$D$的频繁项集。

性质1 给定D，若v为D中的频繁项集，则对任意$w\subseteq v$，都有w是D中的频繁项集。

推论1 给定D，若v为D中的非频繁项集，则对任意$w\supseteq v$，都有w是D中非频繁项集。

2.2 CARMA算法

CARMA算法是一种动态关联规则挖掘方法，用于计算包括来自于网络的事务序列的关联规则。该算法至多需要两次对事务序列的扫描，通过第一次扫描可以得到所有大项集的一个超集以及每个项集支持度的上下边界；在第二次扫描的过程中，该算法通过计算每个大项集的精确支持度，并利用“前向剪枝”技术[16]对所有的非大项集进行修剪。用户在挖掘过程中可以根据实时得到的关联规则对支持度和置信度门限进行调整，如果当前已获得的规则满足要求，用户可以提前停止规则的挖掘。下面分别对该算法的两个阶段进行简要的阐述。

第一阶段即在第一次扫描数据库的过程中，连续地构建一个所有潜在大项集的格(lattice)模型。对于每个项集v都需存储以下3个整数值(图1所示为当前事务索引为i，包含在第j个事务内的项集$\{a,b\}$插入到lattice结构中的情况)。

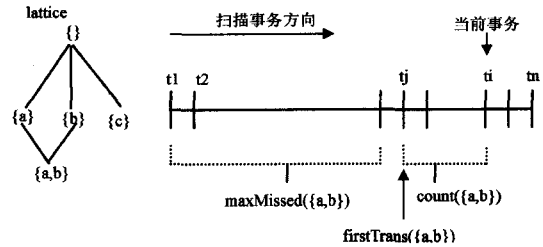

图1 候选项集生成示意图

其中，$count(v)$表示从项集v插入到lattice中开始，v发生的次数；$firstTrans(v)$表示v被插入lattice中所对应的事务的索引；$maxMissed(v)$表示在v插入到lattice之前，v发生次数的上边界。

当扫描事务时增加包含在当前事务中的所有项集的数目。假设当前读取的事务为i，对于任意一个在lattice中的项集v，计算得到关于前i个事务的支持度的下边界$count(v)/i$和上边界$(maxMissed(v)+count(v))/i$，分别用$minSupport(v)$和$maxSupport(v)$表示这两个边界值。在$v$插入lattice的过程中，计算$maxMissed(v)$值是该算法的核心部分，该值的计算不仅取决于$v$和当前事务索引，还与上一个设定的支持度门限相关。在第一阶段对所有事务序列扫描结束后，可以得到一个包含所有大项集超集的lattice。

第二阶段通过逐个读取事务，对lattice中包含的每个项集v重新计算其相关整数$count$和$maxMissed$，然后检查如果$maxSupport(v)$值小于当前用户指定的支持度门限的项集，则把v从lattice中删除。最终得到的lattice为算法的结果，即数据频繁集的集合。算法的详细过程见参考文献[16]。

本文借鉴了CARMA算法至多两次扫描数据库的算法思想，提出了针对大型稠密数据库关联规则挖掘的改进算法VARMLDb。

2.3 DAG结构

VARMLDb算法利用DAG结构替代CARMA算法中的lattice结构存储项集和项集附加信息，如图2所示项集$\{a,b,c,d\}$的$\mathcal{Q}$结构示意图。每个项集存储于$\mathcal{Q}$中的单个节点，并与其前两个子集的结点相连(项集按照字典序排列)，这里用“mother”和“father”分别表示较小子项集和较大子项集。如

$\{a,b\}$和$\{a,c\}$分别是$\{a,b,c\}$的母亲和父亲结点。同时，对于2中的每个项集v还需存储与v的超集的链接，这些保存链接的链表用 childset 表示。

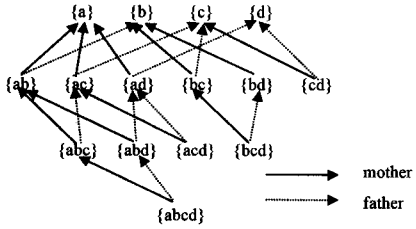

图 2　DAG 结构示意图

2.4　划分 partition 定义

一个数据集的划分$P\subseteq D$定义为包含在数据集D中的所有事务的任意子集，任意两个不同的划分是不重叠的，即$P_i\cap P_j=\varnothing, i\neq j$，且单个划分的大小应与内存大小相匹配。

2.5　垂直数据分布

Apriori 等算法中的数据呈水平分布，其数据集由一系列事务构成，每个事务均有标识符 TID 标识以及相应事务包含相应的项目集。而数据的垂直分布是指，数据集由一系列项目构成，每一个项目均带有与其对应的 TID 列表，即包含该项目的所有事务标识符表 tidlist，且按字典序的升序排列。

2.6　diffset 结构

diffset 与传统的垂直数据分布的不同之处在于，以2结构为基础，每个节点不存储每个项集的 tidlist，而是存储节点的 father 节点和 mother 节点的 tidlist 的差集。如果按传统垂直数据分布，每个节点存储一个项集以及对应的 tidlist，在大型稠密数据库中，这些标识符表往往较大，需要大量的内存存储。而且在进行交集运算时，tidlist 越大运算量就越大。通过 diffset 能极大减少内存使用量和交集运算的计算量。

下面用集合理论的角度对该方法进行阐述。令$t(X)$代表项集X的 tidlist，$d(X)$代表X的 diffset，P为一单项集，$t(P)$为P的 tidlist，PX和PY组合项集，$\sigma(PX)$为PX的支持度。则根据支持度的定义有$t(PX)\subseteq t(P)$以及$t(PY)\subseteq t(P)$，进而通过计算$t(PXY)$的基可以得到项集PXY的支持度，其中$t(PX)\cap t(PY)=t(PXY)$。

现在假设$t(PX)$值未知，但可以得到$d(PX)=t(P)-t(X)$，即项集X和P的 tidlist 之差；类似地可以得到$d(PY)$。显然，组合项集PX的支持度不等于 diffset 的基，而有$\sigma(PX)=\sigma(P)-|d(PX)|$。类似地，现已知$d(PX)$和$d(PY)$，可进一步计算项集$PXY$的支持度$\sigma(PXY)=\sigma(PX)-|d(PXY)|$，则$d(PXY)$的计算方法推导如下：

$$\begin{aligned}d(PXY)&=t(PX)-t(PY)\\&=t(PX)-t(PY)+t(P)-t(P)\\&=(t(P)-t(PY))-(t(P)-t(PX))\\&=d(PY)-d(PX)\end{aligned}$$

显然，计算$d(PXY)$可以不再使用PX和PY的 tidlist 的差值，只需利用PY和PX的 diffset 的差值即可。由此可以说明利用该方法计算频繁项集是正确的。

图 3 为利用 diffset 方法计算频繁项集的过程。其中假设$I=\{a,b,c,d\}$，$T=\{t_1,\cdots,t_6\}$，令$t(v)$表示项集v的 tidlist，$d(v)$表示项集v的 diffset，$v\subseteq I$。

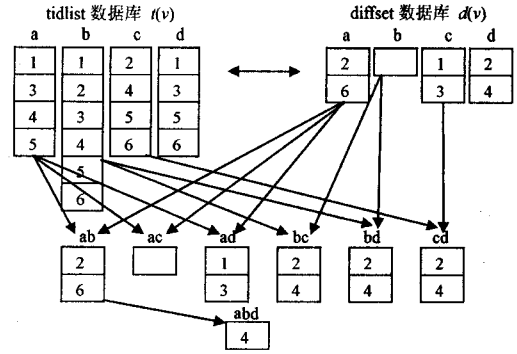

图 3　diffset 方法计算频繁项集

3　改进算法 VARMLDb

3.1　算法思想

在本文提出的算法中，数据集D被划分为n个互不相交的部分$P_1,P_2,\cdots,P_n$，整个挖掘过程完成两次对D的扫描。其中以第一阶段的扫描为算法的核心，形成候选项集的集合2，即所有频繁项集的一个超集；第二阶段工作量较小，不再产生新的候选项集，只负责完成剪枝和输出精确频繁项集的工作。同时，算法保留 CARMA 算法中与项集相关的 5 个值：*count*，*firstTrans*，*maxMissed*，*minSupport*，*maxSupport*。算法详细步骤描述如下。

步骤 1　首先利用单项集的集合对2进行初始化，再通过 ReadPartition 函数对数据库划分中的事务进行读取，同时在此过程中完成2结构的构建、2中单项集 diffset 结构的创建以及2中所有候选项集的 *firstTrans* 和 *maxMissed* 值的计算等工作。

VARMLDb 算法在构建2的过程中通过单项集来构造并计算其他的候选项集，不同于传统的 CARMA 算法在构建 lattice 的过程中，需要检查待插入项集的所有真子集是否已经被插入到 lattice 集合当中，这大大加快了2结构的构建，提高了频繁项集的生成速度。

在单个划分读取完毕之后，利用 UpdateItem 函数对2中候选项集进行迭代计算，更新所有候选项集的 *count*，*minSupport* 和 *maxSupport* 3 个值。

所有划分读取完毕之后，第一次扫描结束，得到包含所有频繁项集的超集结构2。

步骤 2　在第二次扫描之前首先利用条件 *maxSupport*$(v)<$*minsup* 对2进行初步修剪。在之后的扫描数据库过程中，不再增加候选项集个数，对2中项集的 *count* 值和 *maxMisssed* 值进行更新，同时采用 CARMA 算法中的“前向剪枝”技术对2中满足条件的项集进行修剪，并利用 OutPutFreItem 函数对2中的频繁项集进行输出，如果输出的项集在2中没有超集，则将此项集从2中移除，直到2中没有多余频繁项集，则扫描终止。

3.2　算法描述

算法 1　大型稠密数据库关联规则改进算法 VARMLDb

输入：数据库D，单项集集合I，最小支持度 *minsup*。

输出：包含支持度的频繁项集集合F。

//第一次扫描

$2=I$

```
for i=1 to n {//数据库划分个数
    ReadPartition(P_i, 2);
    for each singleton v∈2 {
        v.count+=|D|-|v.diffset|;
        UpdateItem(v);}}
  //第二次扫描
  2=2\{v∈2|maxSupport(v)<minsup};
  for j=1 to m {  //数据库事务数
        ReadTransactions(t_j);
        if 2==null
            exit;
        for all v∈2 {
            ft=v.firstTrans;
            if(v⊆t_j)&&(j<ft)
                v.count++, v.maxMissed--;
        if j==ft {
            v.maxMissed=0;
            v.minSupport=v.maxSupport;
            for all w∈2; v⊂w and
            w.maxSupport>v.maxSupport
                w.maxSupport=v.count-w.count;}
        if v.maxSupport<minsup
            2=2\{v};
        ForwardPrune(v,w,ft);}//前向剪枝函数
    F+=OutPutFreItem(2);
  }
return F;
函数:UpdateItem(V)
输入:2中结点V。
输出:更新V以及V的后代。
for each X∈V.childset {
      X.diffset=X.father.diffset-X.mother.diffset;
      X.count+=X.mother.count-|X.diffset|;}
for each X∈V.childset
      UpdateItem(X);
```

4 实验及其结果分析

本节实验在一台 CPU 主频为 2.8GHz，内存为 1GB，操作系统为 Windows XP 的计算机上进行，编程语言为 Matlab 7.1。为检验算法的优越性，实验选择与 Eclat 算法、Viper 算法和 Mafia 3 种垂直关联算法做比较，其中 Eclat 算法是 Zaki 提出的典型的基于垂直数据分布的关联规则挖掘算法，并结合 tidlist 和前缀树结构，具有较高的执行效率；Viper 算法使用基于压缩垂直位图的方法挖掘关联规则，算法性能在大多数情况下要高于优化的水平算法，即使该水平算法在已知所有频繁项集的情况下，Viper 算法也有较大的优势；Mafia 算法利用 bit-vectors 结构存储频繁项集，使之具有对频繁项集更高的挖掘效率。实验使用的数据库数据由 IBM generator 产生，为了最大程度地模拟真实大型稠密数据，这里生成的项集个数为 130，平均事务长度为 40，共包含 10 000 000 条记录，确保在较高的最小支持度门限下也能产生较长的频繁项集。3 种算法和 VARMLDb 算法的执行结果如图 4 所示，其中最小支持度在 20%～80%之间变化。

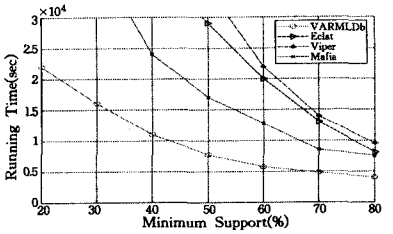

图 4　支持度不同时算法执行时间比较

由图 4 可以看出，在支持度较小时，本文算法的执行时间优势较为明显，而其他 3 种算法的执行时间显然超出了示意图的显示范围。原因是由于较小的支持度门限设置使得在挖掘过程中产生了大量的长候选项集，令 Eclat 算法在为候选项集构造 tidlist 结构时造成庞大的时间开销，而 Viper 算法在挖掘稠密数据集方面性能不如 Eclat 算法，同时，尽管 Mafia 算法本身对存储结构进行了优化，算法效率也优于 Eclat 和 Viper 算法，但从整体执行时间来看，本文算法比其他 3 种算法具有更小的时间复杂度。

图 5 显示的是在最小支持度为 0.5 的基础上，4 种算法在不同记录数下的运行时间对比。在事务数较小时，本文算法 VARMLDb 优势还并不十分明显，但随着事务数的增加，其他 3 种算法在计算长候选项集的 tidlist 时产生了大量中间结果，导致内存逐渐不能满足其要求，需要应用大量的辅存，这带来了内存与磁盘间的页面交换时间消耗。与此相比，此时 VARMLDb 算法中分区和 diffset 结构的优势发挥明显，可以看出，本文算法更适合于大型稠密数据库的关联规则挖掘。

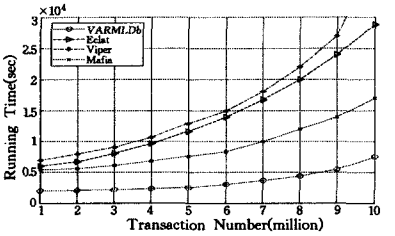

图 5　事务数不同时算法执行时间比较

结束语　本文在前人工作的基础上，提出了一种针对大型稠密数据库的改进算法 VARMLDb。它结合了划分和 diffset 的概念，弥补了传统垂直算法对于大型稠密数据库挖掘能力的不足，在整体上提高了频繁项集的计算效率，减轻了磁盘 I/O 读取开销。实验表明本文算法在性能上优于传统垂直算法，而且最小支持度越小，VARMLDb 算法的优势越明显。本文进一步研究的问题是：改进原有的 maxMissed 的计算公式，使其计算更为简单；研究适合的压缩方法，将要存放在内存中的数据项支持集进行压缩，以减小算法的空间开销，提高算法可扩展性，进一步减少基于数据垂直分布的关联规则挖掘算法运行中频繁项集支持集在内存空间的占用。

参 考 文 献

[1] Agrawa R, Imielinski T, Swami A. Mining association rules between sets of items in large databases[C]// Proc. of ACM SIGMOD International Conference on Management of Date. Washington DC, 1993: 207-216

[2] Park J S, Ming-Syan C, Philip S Y. An Effective Hash Based Algorithm for Mining Association Rules[C]// Proc of ACM

SIGM0D. 1995:175-185

[3] Brin S, Motwai R, Ullman J D, et al. Dynamic Itemset Counting and Implication Rules for Market BasketData[C]// Proc. of ACM SIGMOD Conference on Management of Data. 1997:265-276

[4] Agrawal R, Srikant R. Fast Algorithms for Mining Association Rules in Large Databaes[C]// Proc. of 1994 International Conference on Very Large Databases. 1994:487-499

[5] Savasere S, Omiecinski E, Navathe S. An Efficient Algorithm for Mining Association Rules in Large Databases[C]// Proc. of 21st VLDB. 1995:432-444

[6] Dunkel B, Soparkar N. Data Organization and Access for Efficient Data Mining[C]// Proc. of 15th IEEE Intl. Conf. on Data Engineering. 1999:522-529

[7] Han J, Fu Y J. Mining Multiple-Level Association Rules in Large Database[J]. IEEE Trans. on Knowledge and Data Engineering, 1999, 11(5):798-805

[8] 丁艳辉,王洪国,高明,等. 一种基于矩阵的关联规则挖掘新算法[J]. 计算机科学,2006,33(4):188-189

[9] Shenoy P, Haritsa J R, Sudarshan S, et al. Turbo-charging vertical mining of large databases[C]// Proc. of Intl. Conf. Management of Data. 2000:22-23

[10] Burdick D, Calimlim M, Gehrke J. MAFIA: a maximal frequent itemset algorithm for transactional databases[C]// Proc. of Intl. Conf. on Data Engineering. 2001:443-452

[11] Ayres J, Gehrke J E, Yiu T, et al. Sequential pattern mining using bitmaps representation[C]// Proc. of SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. 2002:429-435

[12] 宋长新,马克等. 改进的 Eclat 数据挖掘算法的研究[J]. 微计算机信息,2008,24(24):92-94

[13] 耿晓斐. 关联规则中 ECLAT 算法的研究与应用[D]. 重庆:重新大学,2009

[14] Zaki M, Parthasarathy S, Ogihara M, et al. New Algorithms for Fast Discovery of Association Rules[C]// Proc. of In 7th International Workshop Research Issues on Data Engineering. 1997:283-286

[15] Poovammal E, Ponnavaikko M. Utility independent privacy preserving data mining on vertically partitioned data [J]. Journal of Computer Science, 2009, 5(9):666-673

[16] Hidber C. Online association rule mining[C]// Proc. of ACM SIGMOD Intl. Conf. on Management of Data. 1999:145-156

(上接第 184 页)

描述可有效帮助驱动开发者避免错误,并可以直接生成用于验证的驱动程序属性描述。Dingo 支持直接编译驱动协议规范,运行时检查器检测驱动程序是否违反协议。

基于上述思路,该大学进一步推出 Termite[7] 系统,以实现驱动程序自动生成。主要思路是:定义操作系统接口规范和硬件接口规范;将定义好的规范形式化,生成接口协议有限自动机;将有限自动机合成为驱动程序。Termite 系统生成的 Linux 驱动程序具有同手工编写程序相同功能和接近性能。Termite 难点在于细化出程序规范,但生成的规范可复用。驱动程序自动生成技术处于试验阶段,是提高驱动程序质量的有益尝试。

结束语 随着系统对可靠性要求的提高,操作系统内核驱动程序可靠性成为国际学术界和工业界关注的热点。从改变操作系统结构的避错和容错手段,到采用程序分析验证技术直接消除驱动程序缺陷纠错手段和采用程序综合技术合成无错设备驱动,驱动程序可靠性增强是一项长期而艰巨的工作。本文从分析驱动程序缺陷入手,指出驱动程序缺陷产生的原因,并以此为线索梳理了目前的解决方法,希望对驱动程序开发和缺陷解决提供借鉴。

参 考 文 献

[1] Engler D, et al. Checking system rules using system-specific programmer-written compiler extensions[C]// Proc. of the 4th USENIX OSDI. Oct. 2000

[2] Coverity Scan Open Source Report [EB/OL]. http://www.coverity.com, 2009

[3] Arthur S. Fault resilient drivers for Longhorn server[R]. Microsoft Corporation, WinHec 2004 Presentation DW04012, May 2004

[4] Witkowski T, et al. Model Checking Concurrent Linux Device Drivers[C]// ASE'07. November, 2007

[5] Kadav A, et al. Tolerating Hardware Device Failures in Software [C]// Proc. of the 22th ACM SOSP. Dec. 2009

[6] Ryzhyk L, et al. Dingo: Taming device drivers[C]// Proc. of the 200 EuroSys Conference. Apr. 2009

[7] Ryzhyk L, et al. Automatic Device Driver Synthesis with Termite[C]// Proc. of the 22th ACM SOSP. Dec. 2009

[8] Ball T, et al. The SLAM project: Debugging system software via static analysis[C]// Proc. of the 29th POPL. 2002

[9] Ball T, et al. Thorough static analysis of device drivers[C]// Proc. of the 2006 EuroSys Conference. 2006

[10] Beyer D, et al. The software model checker BLAST Applications to software engineering[J]. Int J Softw Tools Technol Transfer, 2007

[11] Microsystems S. Solaris Express Software Developer Collection: Writing Device Drivers[M]. chapter 13. 2007

[12] Microsoft Corporation. Introduction to the WDF usermodedriverframework[EB/OL]. http://www.microsoft.com/whdc/driver/wdf/umdf intro.mspx, May 2006

[13] Hunt G, et al. Sealing OS Processes to Improve Dependability and Safety[C]// Proc. of the 2007 EuroSys Conference. 2007

[14] Tanenbaum A. Introduction to MINIX 3 [EB/OL]. http://www.osnews.com/story/15960/Introduction-to-MINIX-3/ Sep 2006 05:41 UTC

[15] Li Man-Lap, et al. Understanding the propagation of hard errors to software and implications for resilient system design[C]// ASPLOS. 2008:265-276

[16] 颜跃进,等. 操作系统设备驱动可靠性研究综述[J]. 计算机工程与科学,2009(5)