一种与缓冲区紧耦合的环形循环滑动窗口的数据流抽取算法

詹 英[1],吴春明[2],王宝军[1]
(1.浙江交通职业技术学院,浙江杭州 311112; 2.浙江大学系统工程研究所,浙江杭州 310027)

摘 要: 许多应用领域产生数据流的流量不断增大,需要挖掘系统提供更有效的数据流抽取策略来应对数据处理能力与流速之间的矛盾.本文构造了基于环形缓冲区循环的逻辑滑动窗口(环形循环滑动窗口),将环形缓冲区和环形循环滑动窗口紧耦合,提出了基于环形循环滑动窗口的数据流抽取算法.实验表明这是适合数据流流动和方便数据流挖掘的数据流抽取模型,能够提升数据流抽取速度和保证数据流的挖掘持续正常实施.

关键词: 数据流; 环形循环滑动窗口; 环形缓冲区; 多线程
中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112 (2011) 04-0894-05

An Algorithm for Data Stream Sampling Based on Ring Circular Sliding Window Tightly-Coupled with Buffer

ZHAN Ying[1], WU Chun-ming[2], WANG Bao-jun[1]
(1. *Zhejiang Institute of Communications, Hangzhou, Zhejiang* 311112, *China*;
2. *Institute of Computer System Architecture, Zhejiang University, Hangzhou, Zhejiang* 310027, *China*)

Abstract: Stream data are generated unboundedly, coping with the conflicts commendably between data processing power and flow velocity may require mining system to provide a more effective solution for Data Stream Sampling. This paper proposed a logical sliding window named ring circular sliding window based on a ring buffer and these two elements were tightly coupled. Then an algorithm for data stream sampling based on ring circular sliding window was proposed. Experimental results show that it is a data stream sampling model which is proper for the migration and mining of data stream, and can achieve further improvement on the speed of data stream sampling and ensure normal execution of data stream mining.

Key words: data stream; ring circular sliding window (RCSW); ring buffer; simultaneous multithreading (SMT)

1 引言

目前许多应用领域在不断地产生大量的数据流,有银行的信用卡业务流、股票交易信息流、零售业务流、电信移动通信数据流、交通实时监控数据流、气象与环境监测数据流等.数据流挖掘与关系数据库知识分析的根本不同在于流数据是快速产生的,数据流隐含着随时间变化的具有很高商业价值的信息,因此从连续产生的,实时到达的,潜在无限的数据流中快速获取知识是企业应对竞争的需要.但快速变化的流数据与有限的内外存空间和处理能力之间的矛盾使得流数据的知识分析变得困难.数据流挖掘系统需要综合研究抽取数据流、分析数据流、反馈知识发现数据流的模式来实现面向高速数据流的知识发现.

2 问题背景

数据流抽样方式大致有水库抽样、直方图、滑动窗口等方法,其中滑动窗口有基于流数据个数定义的和基于时间定义的.目前,有较多的数据流挖掘技术采用滑动窗口完成数据的抽取,Giannella C 等人提出在滑动窗口[1]使用 FP-tree 和 FP-stream 分别维护和挖掘数据流频繁模式的算法.宋宝燕等人提出了基于滑动窗口[2]的支持泛在应用的流聚类挖掘算法 SWBclusting. Hulten G 等人提出基于滑动窗口技术[3]的分类算法 CVFDT.李国微等人提出挖掘数据流任意滑动时间窗口[4]内频繁模式的方法 MSW.虽然很多数据流挖掘算法采用滑动窗口来抽取数据,但针对滑动窗口本身的研究却不多.李康宁等人使用的滑动窗口[5]运行机制是元组存放到缓存当中,待到更新周期到达时将滑动窗口中过期元组淘汰并插入缓存中的元组.基于向量模型的滑动窗口被定义为一个向量[6],根据时间顺序,流数据被推入滑动窗口,新的数据则加入到窗口后端,窗口中的旧数据不断移出,实现数据的流动.李俊奎等人提出了可重写循环滑

收稿日期:2010-07-01;修回日期:2010-09-01
基金项目:浙江省自然科学基金(No.Y1101133);浙江省教育厅项目(No.Y201018946)

动窗口[6],直接计算出即将被移出数据的位置,移入窗口的数据覆盖要移出窗口的数据.Aggarwal 等人在 CluStream 中使用了基于多窗口技术的 pyramidal 时间框架[7].李晓光等提出了基于滑动多窗口[8]的时间序列流趋势变化检测.

由于挖掘系统无法将数据流全部存储起来进行静态数据知识分析,因此数据流的抽取方式决定了参与分析的数据子集,及它的动态变化规律.在滑动窗口中使用衰减因子减少历史数据影响的方法并不能真实描绘数据流的动态变化,从而直接影响数据概念漂移出现的精确判断.在现实商业活动中,数据流的流量并不是稳定的,如超市交易数据流,单位时间内到达的流数据量根据业务量的大小而不断地变化,流速的震荡要求挖掘系统能够提供有效的组织管理策略来应对过载等突变现象.本文从研究数据流抽取方式入手,研究流数据的组织方案,提出基于环形缓冲区循环的逻辑滑动窗口的数据流抽取方式,来提高数据流的处理速度,数据流抽取速度的提高将会提升流数据整体分析速度.

3 相关定义

与传统数据集一样,流数据的类型也有元组数据、事务数据、时序数据等.数据流(Data Stream)连续不断地到达挖掘系统,由于数据流的流速往往大于数据流处理速度,因此规定数据流首先流入缓冲区(Buffer),并根据到达的时间,依序流入滑动窗口(Sliding Window),在滑动窗口中进行数据挖掘的同时,数据流不断地被移出缓冲区,数据由在线状态转入离线状态,并根据历史数据的年龄不断地归档流数据(流数据死亡).

定义 1(流数据的生命周期) 流数据的生命周期存在于流数据挖掘分析的整个运行过程中,我们将流数据的生命周期分为流数据的缓冲期、流数据的窗口期、流数据离线期、流数据归档(死亡)期四个阶段.

定义 2(数据流) 数据流 B 由不断到达缓冲区的流数据组成,是流数据的集合,即 $B=(b_1,b_2,b_3,b_4,b_5,\cdots,b_i,\cdots)$,其中 $b_i(b_{i1},b_{i2},\cdots,b_{im})(i=1,\cdots,n,\cdots)$ 是 m 维流数据.对 $\forall\, b_i(i=1,\cdots,n,\cdots)$ 与 $\forall\, b_j(j=1,\cdots,n,\cdots)$,如果 $i<j$,则 b_i 先于 b_j 到达缓冲区,如果 $j-i=1$,则表示 b_i 到达后,紧跟其后到达的是 b_j.

定义 3(环形缓冲区 RB) 环形缓冲区是一个环形空间结构,由 $Rsize$ 个缓冲区空间单元组成,每个缓冲区空间单元可以存储一条 m 维流数据 $b_i(i=1,\cdots,n,\cdots)$,假设 $Rsize=W$,则缓冲区中每个空间单元的编号依次为 $0,\cdots,W-1$.$RB_j(j=0,\cdots,W-1)$表示在环形缓冲区中,存储在编号为 j 的空间单元的 m 维流数据.

定义 4(环形循环滑动窗口 RCSW) 基于环形缓冲区循环的逻辑滑动窗口简称为环形循环滑动窗口,英文缩写为 RCSW.环形循环滑动窗口逻辑地框定环形缓冲区的一段内存空间,其中存储的流数据为当前瞬间参与分析的数据子集.RCSW 循环不断地沿着一环形缓冲区向前移动,随着数据流在环形缓冲区中的流动,RCSW 内的流数据不断地被排除出 RCSW,新的流数据不断地平滑地被框定到 RCSW 中.数据流挖掘系统只处理在 RCSW 中的流数据,定义滑动窗口大小为 $Wsize$,RCSW 内的流数据集为 $SW=(b_i,b_{i+1},\cdots,b_j)$,$i$ 和 j 是整数,并且 $i<j$,则 $Wsize>=j-i+1$.定义环形循环滑动窗口在环形缓冲区中的数据入口位置编号为 $WindowIn$,数据出口位置编号为 $WindowOut$,则在流数据集 SW 中,流数据 b_i、b_j 分别存储在编号为 $WindowOut$、$WindowIn$ 的环形缓冲区空间单元中.

4 环形循环滑动窗口

4.1 数据流的生命周期

环形循环滑动窗口全景图如图 1 所示,RCSW 平滑地逻辑地融入到环形缓冲区中.在 Buffer 区域中的流数据处于缓冲期,等待被处理;指针标记 $Head$ 是数据流入缓冲区的入口.在 RCSW 区域中的流数据处于窗口期,正在被处理.通过窗口起始标记 $WindowIn$,窗口尾端标记 $WindowOut$,将环形缓冲区中的一段区域标识为环形循环滑动窗口 RCSW.环形缓冲区与 RCSW 之间紧密耦合.流数据的生命周期从流数据的缓冲期变迁到流数据的窗口期,其间不发生任何数据移动.

图 1(a)显示,在缓冲区中,编号为 0 到 i 空间区域被标识为环形循环滑动窗口区域,图 1(b)显示有新的流数据流入到缓冲区 $Head$ 标识的空间单元,此时,在

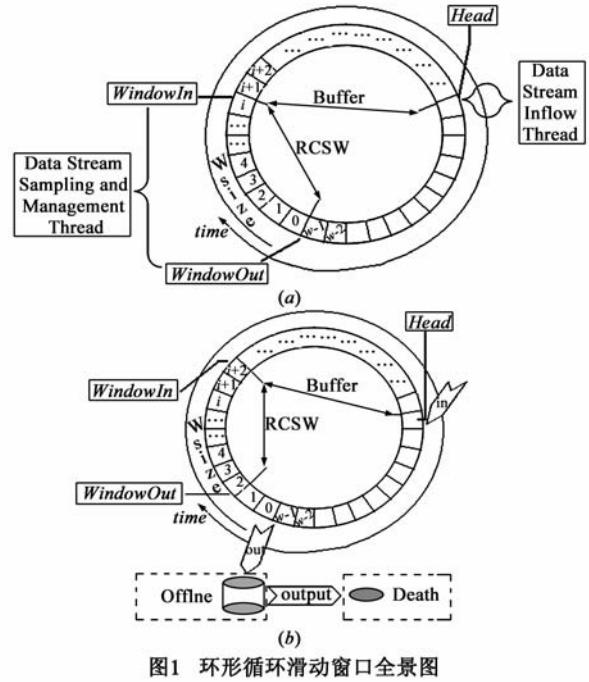

图1 环形循环滑动窗口全景图

缓冲区中,编号为 2 到 $i+2$ 空间单元被标识为环形循环滑动窗口区域.

在挖掘处于窗口期的流数据的同时,流数据源源不断地流入环形缓冲区.被分析后的数据可以根据需要被分流到外存.当流数据处于离线期时,可以对其进行传统数据挖掘,借助离线挖掘结果来判断在线数据流挖掘的精度变化,反过来影响在线数据流的挖掘模式的调整.最后根据需要归档或删除历史流数据,流数据死亡.

4.2　面向环形缓冲区的多线程并发控制

为流数据不同生命周期的处理任务分配线程,并发控制数据流,获得数据流挖掘性能的提升.本文主要给出数据流流入环形缓冲区线程和 RCSW 数据流抽取与管理线程.下面首先给出环形循环滑动窗口 RCSW 和环形缓冲区 *RB* 的形式化定义.

$RCSW = (Wsize, WindowIn, WindowOut, MoveStep)$;

$B = (Rsize, Head, empty, full)$,

则 $SW = (RB_{WindowOut}, \cdots, RB_{WindowIn})$,其中 $RB_k (k = 0, \cdots, Rsize - 1)$存储了 m 维流数据 b_i.定义 *BI* 是 m 维流数据类.

其中 *Wsize* 表示滑动窗口大小;*MoveStep* 表示环形循环滑动窗口移动步长;*SW* 表示窗口内参与分析的流数据集;*Rsize* 表示当前环形缓冲区的单元数;*head* 表示新数据流入环形缓冲区的入口位置;*empty* 表示空的缓冲区单元数;*full* 表示等待处理的缓冲区单元数.对象类 *RingBuffer* 由环形缓冲区对象 *Rb*、环形循环滑动窗口对象 *Rcsw*、*Sampling* 方法、*Inflow* 方法构成.线程 *Sample* 调用 *RingBuffer* 对象中的 *Sampling*()方法获取流数据,线程 *InFlow* 调用 *RingBuffer* 对象中的 *Inflow*()方法向环形缓冲区 *Rb* 放入流数据.多线程并发控制示意图如图 2 所示.

在 RCSW 中,数据流流动过程示意图如图 3 所示,在初始状态,环形缓冲区与环形循环滑动窗口的定义如下:

$Rcsw = (Wsize = 5, WindowIn = 0, WindowOut = 0, MoveStep = 2)$;

$Rb = (Rsize = 150000, Head = 0, empty = 150000, full = 0)$.

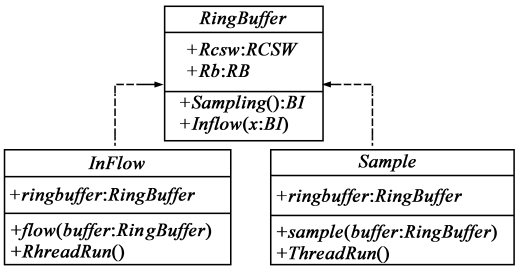

图2　多线程并发控制示意图

当流数据 b_1、b_2、b_3 流入缓冲区时,RCSW 未满,此时 $WindowIn = 2$, $Head = 2$, $WindowOut = 0$.当流数据 b_4、b_5、b_6、b_7、b_8 流入缓冲区时,RCSW 已满,此时 $WindowIn = 4$, $WindowOut = 0$, $Head = 7$.挖掘处于窗口期的流数据,并可同时复制窗口期数据到外存.当流数据 b_9 流入缓冲区时,系统处理完毕处于窗口期的流数据,并判断滑动窗口外是否还有流数据,如果足够,系统推动 RCSW 向前移动一个步长位置,部分数据从 RCSW 中移出,此时 $WindowIn = 6$, $WindowOut = 2$, $Head = 8$.同时在挖掘模式中删除流数据 b_1、b_2 后,将新加入窗口的流数据 b_6、b_7 添加到挖掘模式.

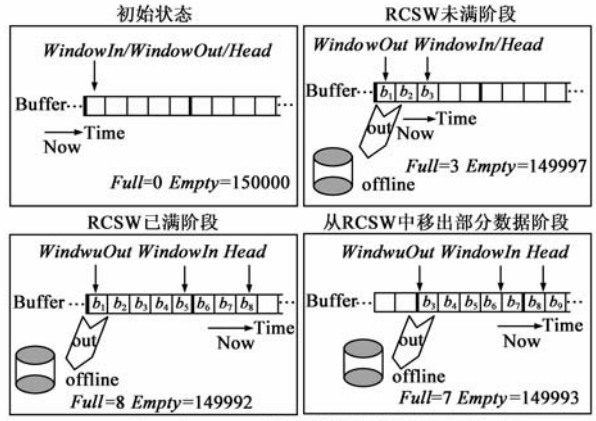

图3　在RCSW中数据流流动过程示意图

4.3　RCSW 的特点

RCSW 结构简单清晰,用首尾标识逻辑地框定环形缓冲区中的一段空间为滑动窗口,环形缓冲区与滑动窗口紧密耦合,通过调整窗口起始标记 *WindowIn*、窗口尾端标记 *WindowOut*,滑动窗口只需向前不断移动,这使得 RCSW 移动滑动窗口的时空代价很少,具有如下特点.

(1)增量性:基于 RCSW 的流数据抽取是增量的.假设当前 $SW = (RB_{WindowOut}, \cdots, RB_{WindowIn})$,定义 $x = (WindowIn + 1) \% Rsize$, $y = (WindowIn + MoveStep) \% Rsize$, $z = (WindowOut + MoveStep - 1) \% Rsize$,则移动一个步长后,新增流数据集合 $Add = (RBx, \cdots, RBy)$,需要移除的流数据集合 $Move = (RB_{Windowout}, \cdots, RBz)$.

(2)可控性:滑动窗口和缓冲区大小可控、移动步长可控.如果数据流速度加快,可以通过改变 *Rsize* 属性增加环形缓冲区的空间,调整 *MoveStep* 来控制移动步长,调整 *WindowIn* 与 *WindowOut* 适当加大窗口大小来加快数据处理,当数据流速异常加速时,可以将 *WindowIn* 调整到 *Head* 处,来实现滑动窗口的跳跃.例如,在股票交易时,用户只关心最新交易情况,我们可以将滑动窗口跳跃到最新流数据位置,通过忽略部分历史数据流来缓解处理速度跟不上流速的矛盾.

(3)可组合性:组合 RCSW,通过一个 RCSW 控制一

条数据流的方式来接收多数据流.

4.4 数据流控制算法

数据流流入环形缓冲区算法:判断环形缓冲区是否有足够的空间,如果空间足够,将流数据推入环形缓冲区 *Rb*.

输入:$b_i(i=1,\cdots,n,\cdots)$, *ringbuffer*
输出:接收了流数据的环形缓冲区 *Rb*
putinbuffer(b_i);//将 b_i 流数据推入 *Rb*
Rb.*Head* = (*Rb*.*Head* + 1)%*Rsize*;//调整 *Rb* 入口
Rb.*empty* = *Rb*.*empty* − 1;//调整 *Rb* 空闲单元数
Rb.*full* = *Rb*.*full* + 1;//调整 *Rb* 单元流数据数

RCSW 数据流抽取与管理算法:逻辑变量 *flag* 标识当前 RCSW 内数据是否已满,如果已满,则判断环形缓冲区是否有足够的流数据,如果足够,逻辑地推动 RCSW 沿着环形缓冲区 *Rb* 向前移动.

输入:*ringbuffer*
输出:挖掘模式或 $b_i(i=1,\cdots,n,\cdots)$
if (*flag*)
{while (*full* > *Rcsw*.*MoveStep*)
{for (*int i* = 0; *i* < *Rcsw*.*MoveStep*; *i* ++)
{*deletemodel* (*Rcsw*.*WindowOut*);
//从挖掘模式中删除移出 *Rcsw* 的流数据
Rcsw.*WindowOut* = (*Rcsw*.*WindowOut* + 1)%*Rsize*;//调整 RCSW 尾端位置}
for (*int i* = 0; *i* < *Rcsw*.*MoveStep*; *i* ++)
{*Rcsw*.*WindowIn* = (*Rcsw*.*WindowIn* + 1) % *Rsize*;//调整 RCSW 前端位置
insertmodel (*Rcsw*.*WindowIn*);
//从挖掘模式中添加移入 *Rcsw* 的流数据}
Rb.*empty* = *Rb*.*empty* + *Rcsw*.*MoveStep*;
Rb.*full* = *Rb*.*full* − *Rcsw*.*MoveStep*;
//调整 *Rb* 空闲单元数和流数据数}}

5 实验及分析

实验环境:CPU 为 Pentium Dual-Core 2.0GHZ、内存为 2GB;操作系统为 Windows XP;采用 *c*# 实现各类滑动窗口.模拟数据流是随机产生的一维平稳数据流.实验中使用并发控制机制实现数据流持续流入缓冲区与数据流处理的同步运行.数据流处理环节被简化为接收流数据.

实验一主要目标是获取在滑动窗口大小(*Wsize*),滑动窗口移动步长(*MoveStep*)取不同值情况下,基于向量模型的滑动窗口(Vector)与可重写循环滑动窗口(Circular)的数据处理时间.上述两类滑动窗口的数据处理时间对照图如图 4 所示,时间单位为毫秒(ms).从图 4 可以看出,可重写循环滑动窗口的数据处理速度比向量模型滑动窗口的数据处理速度快,但随着滑动窗口移动步长和滑动窗口大小的增大,这种速度上的优势会迅速减弱.在滑动窗口大小相同的情况下,上述两类滑动窗口的运行时间随着移动步长的增大而减少.在滑动窗口移动步长相近时,上述两类滑动窗口的运行时间随着滑动窗口大小的增大而减少.

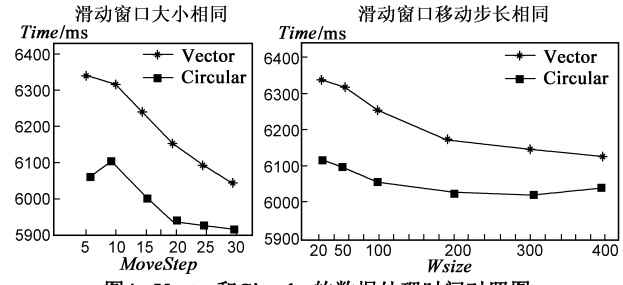

图4 Vector和Circular的数据处理时间对照图

当前数据流的速度不断加快,流量不断增大,流速也不断地震荡.太小的窗口使得同时参与分析的数据量也少,分析结果将不能准确表达数据流的动态变化规律,从而直接影响数据概念漂移出现的精确判断.太小的移动步长显然也不能解决数据处理能力与流速、流量与有限空间之间的矛盾.当在数据流速异常加快时,这使得流数据的知识分析变得更加困难,因此需要研究适合的数据流管理模型,使在一定的 CPU、内存资源条件下,数据流的挖掘能够持续正常实施.

实验二主要目标是获取在滑动窗口大小取不同值情况下,考查环形循环滑动窗口(RCSW)与可重写循环滑动窗口(Circular)的数据处理时间.所有时间取值是在相应条件下 5 次实验值的平均值.RCSW 和 Circular 的数据处理时间对照图如图 5 所示,可以看出,环形循环滑动窗口的数据处理速度大大快于可重写循环滑动窗口的数据处理速度.在滑动窗口大小不同的情况下,环形循环滑动窗口的数据处理速度比较稳定.

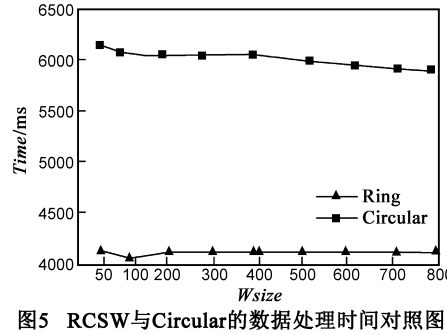

图5 RCSW与Circular的数据处理时间对照图

在平稳数据流的情况下,增加实验样本数,来分析比较三类滑动窗口在同一数据流上的处理器运行性能,如图 6 所示,样本数分别为 4000 和 8000.图 6 中的第Ⅰ条、第Ⅱ条、第Ⅲ条粗线波形分别是运行向量模型滑动窗口、可重写循环滑动窗口、环形循环滑动窗口而形成的;贯穿整个运行时间的细线波形是系统总的处理器时间.

从图 6 可见,环形循环滑动窗口占用总的处理器时

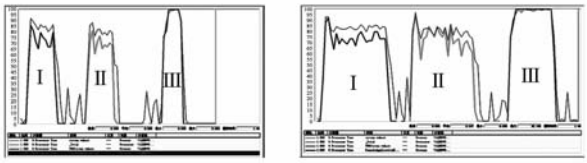

图6　三类滑动窗口处理器时间比较(4000\8000样本)

间最短,而且随着样本数的增加,数据流抽取速度的优势更加明显.在数据流处理运行过程中,环形循环滑动窗口占用总的处理器时间比可重写循环滑动窗口和向量模型滑动窗口稳定,波动少.随着内存空间的不断增大,基于 RCSW 的数据流挖掘可以用宽裕的空间来换取数据流挖掘效率的提高.因此 RCSW 更适合为具有大数据量、流速快速震荡等特点的数据流进行数据处理.

6　结束语

在数据流挖掘领域,数据流抽取模式一直是研究的热点,通过上述实验,我们可看到,环形循环滑动窗口与环形缓冲区高度耦合,基于环形循环滑动窗口的数据处理速度大大优于其他两类滑动窗口的数据流抽取模式.实验表明基于环形循环滑动窗口的数据流抽取模式是适合数据流流动和方便数据流挖掘的,它能够保证数据流的挖掘持续正常实施.由于数据流上的数据持续到达,流速是可变的,不可预知,系统的承载能力随着流速的变化而改变,因此能够更好地适应不可知的数据流速的数据流抽取方式才具有较好的稳定性和可靠性.针对这一问题,将延伸研究数据流的速度异常突变检测与应对策略和数据流的智能驱动方案.

参考文献

[1] Giannella C, et al. Mining frequent patterns in data streams at multiple time granularities[A]. Data Mining: Next Generation Challenges and Future Directions[M]. Sri Lanka: AAAI/MIT Press, 2004. 191 – 212.

[2] 宋宝燕,等.基于滑动窗口的支持泛在应用的流聚类挖掘算法[J].小型微型计算机系统,2008,29(12):2262 – 2267.
SONG Bao-yan, et al. Clustering algorithm for ubiquitous data stream mining based on slip window[J]. Journal of Chinese Computer Systems, 2008, 29(12): 2262 – 2267. (in Chinese)

[3] Hulten G, et al. Mining time-changing data streams[A]. Proc. of the 7th ACM SIGKDD[C]. USA: ACM Press, 2001.

[4] 李国微,等.挖掘数据流任意滑动时间窗口内频繁模式[J].软件学报,2008,19(10):2585 – 2596.
LI Guo-Hui, et al. Mining the frequent patterns in an arbitrary sliding window over online data streams[J]. Journal of Software, 2008, 19(10): 2585 – 2596. (in Chinese)

[5] 李康宁,等.一种数据流滑动窗口范围连接上基于局部特征的查询索引[J].计算机研究与发展,2006,43(Suppl.):161 – 167.
Li Kangning, et al. Locality-aware query index for range join on sliding window over data stream[J]. Journal of Computer Research and Development, 2006, 43(Suppl.): 161 – 167. (in Chinese)

[6] 李俊奎,等.可重写循环滑动窗口:面向高效的在线数据流处理[J].计算机科学,2007,34(12):51 – 55.
LI Jun-kui, et al. Rewritable circular sliding window: Towards effective on-line data stream processing[J]. Computer Science, 2007, 34(12): 51 – 55. (in Chinese)

[7] Aggarwal CC, et a1. A framework for clustering evolving data streams[A]. Proc of the 29th VLDB Conference[C]. Berlin: Morgan Kaufmann Publishers, 2003. 81 – 92.

[8] 李晓光,等.基于滑动多窗口的时间序列流趋势变化检测[J].电子学报,2010,38(2):321 – 326.
Li Xiao-guan, et al. Sliding multi-windows based trend change detection on time series stream[J]. Acta Electronica Sinica, 2010, 38(2): 321 – 326. (in Chinese)

[9] Babcock B, et al. Models and issues in data stream systems [A]. Proc. of the 21st ACM Symposium on Principles of Database Systems[C]. USA: ACM Press, 2002. 1 – 16.

[10] 沈铖,孙义和.一种支持同时多线程的 VLIW DSP 架构[J].电子学报,2010,38(2):352 – 358.
SHEN Zheng, SUN Yi-he. Architecture design of simultaneous multithreading VLIW DSP[J]. Acta Electronica Sinica, 2010, 38(2): 352 – 358. (in Chinese)

[11] 姜明,等.网络流量预测中的时间序列模型比较研究[J].电子学报,2009,37(11):2353 – 2357.
JIANG Ming, et al. Research on the comparison of time series models for network traffic prediction[J]. Acta Electronica Sinica, 2009, 37(11): 2353 – 2357. (in Chinese)

[12] 钱江波,等.数据流窗口连接查询处理器研究[J].电子学报,2009,37(2):404 – 409.
QAN Jiang-bo, et al. Hardware processor for window joins over multiple data streams[J]. Acta Electronica Sinica, 2009, 37(2): 404 – 409. (in Chinese)

作者简介

詹　英　女,1970 年生于浙江临海,硕士,目前为浙江交通职业技术学院副教授,主要研究方向为数据库技术与数据挖掘.
E-mail: zhanying@zjvtit.edu.cn

吴春明　男,1967 年生于浙江萧山,博士,浙江大学计算机学院教授、博士生导师,主要研究方向为网络服务质量、可重构网络技术与网络虚拟化、数据挖掘与共享技术.
E-mail: wuchunming@zju.edu.cn