基于 RCSW 的数据流速度异常检测算法研究

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: 目前许多应用领域产生数据流的流速不断地震荡,使得面向数据流的挖掘变得困难.系统采用 RCSW 来完成数据流抽取,提出了实时度 T、关键时点集、数据流处理率的概念,并进一步提出了数据流速度异常检测算法.系统监控、预测数据流速,当数据流速异常减速或增速时,系统智能调节环形缓冲区和数据流处理率来应对异常,为解决数据流处理能力与流速、流量与有限空间之间的矛盾提供解决方案.实验表明数据流速度异常检测算法能够保证数据流的挖掘持续正常实施,最大程度的满足系统的实时性要求.

关键词: 数据流; 环形循环滑动窗口; 关键时点; 实时度 T
中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112-(2012)04-0674-07
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2012.04.009

An Algorithm for Data Stream Speed Anomaly Detection Based on RCSW

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: In many fields, data stream continues to grow in terms of generation speed, scale and vibration, which makes the data stream mining more difficult. RCSW is used in data stream mining to finish data steam sampling. The three concept such as real-time T, key time point set, data stream processing ratio are proposed. Then an algorithm for data stream speed anomaly detection is proposed, which monitor and predict flow velocity. The system intelligently adjust ring buffer and data stream processing ratio if there is excessive flow velocity, in order to solve the conflicts commendably between data processing power and flow velocity, throughput and limited memory. Experimental results show that it is an algorithm for data stream speed anomaly detection which can ensure normal execution of data stream mining and well meet the need of the system real-time.

Key words: data stream; ring circular sliding window (RCSW); time point; real-time T

1 引言

在现实商业活动中,流数据持续地高速地到达,单位时间内的数据流量根据业务量的大小而不断地变化,流速不可预知.数据流速的震荡要求挖掘系统能够智能地适应不可知的数据流速;能够实时地返回处理结果.但当数据流速大于数据流处理能力时,在有限的内存空间完成无限的数据流的精确分析变成不可能完成的任务.面对无限制随机高速流入的数据流,系统无法在有限的存储空间上记录全部信息,因此现实可行的办法是实时地获得尽可能准确的挖掘结果.在震荡数据流上监测流速,并根据流速与数据流处理速度,及时做出合理的应对策略,最大化数据流抽取率和处理能力.因此提高数据流抽取速度和抽取率,最终提高数据流挖掘精度成为一项具有挑战性的任务.

国内外对数据流抽样技术进行了广泛的研究,Vitter[1]提出了水库抽样,随着数据流的流动,随机地从"水库"中抽取 S 个样本元素,水库抽样中所有流数据具有相同的权值,Gibbons[2]等人提出了计数抽样方法,能够抽取出大概率样本.很多数据流挖掘算法采用滑动窗口来抽取数据,Giannella C[3]等人提出在滑动窗口使用 FP-tree 分别维护和挖掘数据流频繁模式的算法.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.但上述算法均未涉及到当流速不断变化时,系统如何智能地调整数据流抽取方案以适应流速的震荡,从而提高数据流挖掘准确性.笔者在文献[5]中提出了基于环形循环滑动窗口 RCSW 的数据流抽取模式.随着数据流在环形缓冲区中的流动,RCSW 内的流数据不

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)

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.

环形循环滑动窗口 RCSW 和环形缓冲区 RB 的形式化定义如下[5].

RCSW = (Wsize, WindowIn, WindowOut, MoveStep)

RB = (Rsize, Head, empty, full)

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;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.WindowIn、WindowOut 分别表示 RCSW 起始标记和尾端标记.如图 1 所示,环形循环滑动窗口 RCSW 循环不断地沿着环形缓冲区向前移动.

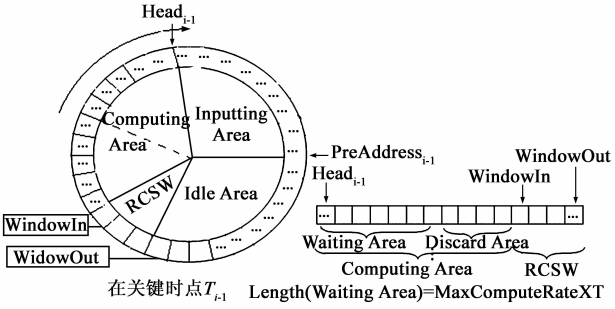

图1　环形缓冲区四个区域示意图

实验证明环形循环滑动窗口 RCSW 是适合数据流流动和方便数据流挖掘的.在此基础上,本文重点给出基于 RCSW 的数据流速度异常检测算法,通过预测数据流速,智能调节环形缓冲区和数据流处理率来应对突变现象.

2　相关定义及问题描述

2.1　相关定义

定义 1　用时间 T 来度量数据流挖掘系统对数据流处理的实时性要求,实时度 T 表示系统要求新产生的数据流在 T 时间内处理完毕并反馈处理结果.

定义 2　关键时点集 KeyTSet = $\{T_1、T_2、\cdots、T_{i-1}、T_i、T_{i+1}、\cdots\}$是一个无限的关键时间点序列,在每一个关键时间点 $T_i(i=1,2\cdots)$,系统监测数据流,计算从 T_{i-1}到 T_i 时间段的数据流的实际流速 $Realrate_{i-1}$、实际数据处理速度 $RealComputeRate_{i-1}$.

定义 3　定义实际数据流处理速度集(…、$RealComputeRate_{i-1}$、$RealComputeRate_i$、…)中的最大值为最大数据流处理速度 MaxComputeRate;其中,$RealComputeRate_i$ 为 $T_i\cdots T_{i+1}$时段的实际数据流处理速度.

定义 4　数据流挖掘过程中,RCSW 不断地沿着环形缓冲区转动.环形缓冲区中的每一个存储单位不断地在“空闲”、“等待流入”、“等待处理”、“正在处理”四种状态中变迁.如图 1 所示,将环形缓冲区分为“空闲区”(Idle Area)、“流入区”(Inputting Area)、“处理区”(Computing Area)、“RCSW”四个区域,“处理区”可拆分成“待处理区”(Waiting Area)、“忽略区”(Discard Area).

在每一个 T_i 关键时点,系统重新定义 $T_i\cdots T_{i+1}$时段的四个区域,在环形缓冲区,RCSW 的入口位置到环形缓冲区数据流入口位置 Head 之间的区域被定义为“处理区”.“处理区”中的存储空间包含了所有没被处理的流数据.当“处理区”的长度超过 MaxComputeRate × T 时,“处理区”拆分成“待处理区”、“忽略区”,其中规定“待处理区”的长度为 MaxComputeRate × T.使得“待处理区”内的流数据能够满足实时度 T 的约束,“忽略区”中的数据流将不会参与数据分析.当“处理区”的长度小于 MaxComputeRate × T 时,在“处理区”中的所有流数据均能够满足实时度 T 的约束,此时“处理区”只包含“待处理区”.

在 T_i 关键时点,如果 $Realrate_{i-1}$ > MaxComputeRate,定义从环形缓冲区数据流入口位置 Head 开始长度为 $Realrate_{i-1} \times T$ 的区域为“流入区”.系统估计在 T_{i+1}关键时点,“流入区”将被占满.当 $Realrate_{i-1} \leqslant$ MaxComputeRate 时,定义从环形缓冲区数据流入口位置 Head 开始长度为 MaxComputeRate × T 的区域为“流入区”.

在环形缓冲区中,除去“流入区”、“处理区”、“RCSW”三个区域空间后剩下的空间被定义为“空闲区”.

定义 5　当环形缓冲区中的处理区内包含“忽略区”时,系统处于降载状态,否则系统处于正常状态.

定义 6　定义最小环形缓冲区 MinBuffer 的存储单元数为 3 × MaxComputeRate × T + Wsize.

合理性　当数据流速 $Realrate_i$ 小于数据流最大处理能力时,系统能够及时处理所有流数据,并等待流数据的流入.此时,数据流的实际处理速度 $RealComputeRate_i$ 将小于等于 $Realrate_i$,当数据流速 $Realrate_i$ 很慢并趋向于零时,“处理区”空间大小也将趋于零,因此系统为“处理区”、“空闲区”、“流入区”的空间留足 MaxComputeRate × T,以适应数据流的突然加速和实现最大处理能力.

定义 7　在每一个关键时点 T_i,定义 $T_i\cdots T_{i+1}$时段预测速度 $Prerate_i$ 为 $Realrate_i$,需要空闲存储单元标准值为 2 × max($Prerate_i$, MaxComputeRate) × T.

合理性　根据定义 4,在 T_i 关键时点,“处理区”中框定了所有没被处理的流数据.RCSW 框定了所有正被处理的流数据,此时此刻,环形缓冲区中其余存储单元均为空,系统预估 $T_i\cdots T_{i+1}$时段的“流入区”空间大小为 max($Prerate_i$, MaxComputeRate) × T, 当 $Realrate_i \leqslant$ max

($Prerate_i$, MaxComputeRate)时,“流入区”空间足够存储用来从 T_i 关键时点开始,T 时间内新产生的流数据,但是当 $Realrate_i > n \times \max(Prerate_i, MaxComputeRate)$ ($n \geqslant 1$, n 是整数)时,“流入区”空间不足用来存储 T 时间内新产生的流数据,此时,系统至少需要 $n \times \max(Prerate_i, MaxComputeRate) \times T$ 空间用来存储 T 内新产生的流数据,所以只能占用“空闲区”的空间.但系统处理流数据的能力有限,在 T 内,系统最多能够处理 MaxComputeRate $\times T$ 空间大小的流数据.在数据流速异常加快,并迅速占满“流入区”时,系统需要迅速反应,并且依旧需要存储空间来接受新的流数据,否则流数据将可能溢出,为了保证系统能够持续正常运行,定义“空闲区”空间至少为 $\max(Prerate_i, MaxComputeRate) \times T$.因此,在每一个关键时点 T_i, $T_i \in$ KeyTSet,定义 $T_i \cdots T_{i+1}$时段需要空闲存储单元标准值为 $2 \times \max(Prerate_i, MaxComputeRate) \times T$,只要空闲存储单元大于等于标准值,表示系统已经预留了足够空间与时间用来应对数据流速异常加快的现象.

定义 8 在 $T_i \cdots T_{i+1}$时段,流数据的处理率 $P_i = RealComputeRate_i / RealRate_i$.流数据的全程处理率 $P = (\sum P_i)/n$.

2.2 实时度 T 框架模型

由于数据流的持续流动性,流速的不可预知性,而数据流挖掘在实际应用中强调数据知识分析的及时性、实时性,因此需要建立面向数据流的,支持实时度约束的实时度 T 框架模型.实时度 T 框架模型挖掘最近的,并在实时度 T 内能够处理完毕的流数据,直接忽略那些肯定无法满足实时度约束的流数据.在每一个关键时点 T_i, $T_i \in$ KeyTSet,系统执行如下任务.

(1)检测时段 T_{i-1}到 T_i 数据流实际流速 $Realrate_{i-1}$ 和数据流实际处理速度 $RealComputeRate_{i-1}$,预估下一时段 T_i 到 T_{i+1}的数据流预测流速 $Prerate_i$ 和数据流预测处理速度 $PreComputeRate_i$,其中,$Prerate_i = Realrate_{i-1}$, $PreComputeRate_i = RealComputeRate_{i-1}$;

(2)预估在关键时点 T_i,“流入区”需要的空间 $PreSpace_i$,其中,$PreSpace_i = \max(Prerate_i, MaxComputeRate) \times T$;

(3)在关键时点 T_i 到时点 $T_i + T + \delta$ 之间产生新的关键时点 T_{i+1}.关键时点 $T_{i+1} = T_i + T'$,其中,$0 < T' \leqslant T + \delta$($\delta$ 是允许的时间误差).

如果 $Realrate_i > \max(Prerate_i, MaxComputeRate)$,则从 T_i 关键时点开始过了 T'($T' < T$)时间,“流入区”空间被全部占满.此时:

$$T' = (\max(Prerate_i, MaxComputeRate)/Realrate_i) \times T$$

如果 $Realrate_i \leqslant \max(Prerate_i, MaxComputeRate)$,则从 T_i 关键时点开始过了 $T + \delta$ 时间,“流入区”空间刚被或未被全部占满.此时:$T' = T + \delta$.

3 数据流监控算法

RCSW 结构简单清晰,环形缓冲区与 RCSW 紧密耦合,这使得 RCSW 具有增量性和可控性等特点[5].RCSW 与环形缓冲区大小可控、移动步长可控.调整 RCSW 的入口与出口位置,可以实现 RCSW 的跳跃.本节介绍基于 RCSW 的面向流速震荡的数据流速度检测算法和数据流控制策略.

3.1 满足实时度 T 要求的数据流过载检测算法

性质 1 当实际数据流速度不断增大,并最终趋于稳定,且,Realrate > MaxComputeRate,此时,环形缓冲区的存储单元数也逐渐增大,并最终趋于稳定,此时“处理区”、“空闲区”与“流入区”的长度也保持等于 Realrate $\times T$,由于流数据流速大于系统最大处理能力,“处理区”将包含“忽略区”,即,处理率 P 小于 100%.

下面通过一个具体的例子来说明环形缓冲区的变化.假设,$MaxComputeRate < Realrate_{i-2} < Realrate_{i-1} = Realrate_i$.

在关键时点 T_{i-1},假设“处理区”存储单元数为 $Realrate_{i-2} \times T$,环形缓冲区空间总数为 $3 \times Realrate_{i-2} \times T + Wsize$.系统预测 $T_{i-1} \cdots T_i$ 时段流速为 $Realrate_{i-2}$,预估“空闲区”与“流入区”存储单元数均为 $Realrate_{i-2} \times T$,此时环形缓冲区满足空闲存储单元标准值要求,关键时点 T_{i-1}状态示意图如图 1 所示.

在关键时点 T_i,预测 $T_i \cdots T_{i+1}$时段流速为 $Realrate_{i-1}$,此时“处理区”存储单元数为 $Realrate_{i-1} \times T' + MaxComputeRate \times (T - T')$($T' = Realrate_{i-2} \times T / Realrate_{i-1}$),预估“空闲区”与“流入区”需要存储单元数均为 $Realrate_{i-1} \times T$,环形缓冲区不满足空闲存储单元标准值要求,需要将空闲存储单元数增加到 $2 \times Realrate_{i-1} \times T$.增加空间后,环形缓冲区空间总数为 $2 \times Realrate_{i-1} \times T + Realrate_{i-1} \times T' + MaxComputeRate \times (T - T') + Wsize$,由于,$MaxComputeRate < Realrate_{i-1}$,且 $T' < T$,所以,此时环形缓冲区空间总数小于 $3 \times Realrate_{i-1} \times T + Wsize$,流速增速时,关键时点 T_i 状态示意图如图 2 所示.

在关键时点 T_{i+1},“处理区”存储单元数为 $Realrate_i \times T$($Realrate_{i-1} = Realrate_i$).预测 $T_{i+1} \cdots T_{i+2}$时段流速为 $Realrate_i$,预估“空闲区”与“流入区”需要存储单元数均为 $Realrate_i \times T$,环形缓冲区不满足空闲存储单元标准值要求,需增加空间到 $2 \times Realrate_i \times T$,增加空间后,环形缓冲区空间总数为 $3 \times Realrate_i \times T + Wsize$.流速增速时,关键时点 T_{i+1}状态示意图如图 3 所示.

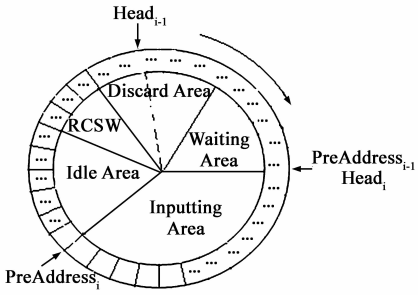

图2　流速增速时,关键时点T_i状态示意图

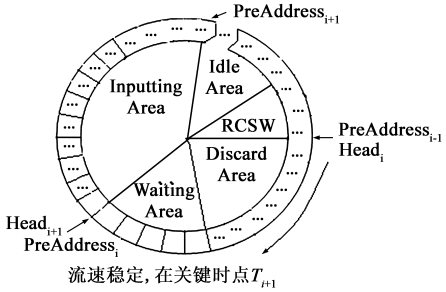

图3　流速增速时,关键时点T_{i+1}状态示意图

性质 2 当实际数据流速度是稳定的,并且等于最大数据流处理速度时,即,Realrate = MaxComputeRate,此时"空闲区"与"流入区"的长度也保持等于 MaxComputeRate× T,环形缓冲区的空间单元数也将稳定等于最小值,即,3 × MaxComputeRate × T + Wsize. 此时,所有流入环形缓冲区的流数据均被处理,"处理区"将不包含"忽略区",即,处理率 P 的值约等于 100%.

引理 1　在满足实时度 T 的约束情况下,在关键时点 T_{i+1}(Prerate$_i$ = Realrate$_{i-1}$),当环形缓冲区的各参数满足以下条件时,证明降载或增加空间的必要性.

条件 1　Length(Buffer) ≥ (2 × Max(Prerate$_i$, MaxComputeRate) + MaxComputeRate) × T + Wsize;

条件 2　Realrate$_i$ > MaxComputeRate;

Prerate$_i$ > MaxComputeRate;

full$_i$ ≥ MaxComputeRate × T;

条件 3　不处于降载状态;

条件 4　empty$_{i+1}$ < 2 × Realrate$_i$ × T;

证明　当"处理区"的空间数大于 MaxComputeRate × T,表示有多余的流数据不能满足实时度 T 的约束,需要丢弃这些多余的不能及时处理的流数据. 因此需要通过降载,保证"处理区"中的流数据能够在 T 时间内全部处理完毕. 因此,要证明需要降载,只需证明"处理区"的空间数 Full$_{i+1}$ > MaxComputeRate × T.

情况 1　假设 Realrate$_i$ ≤ Prerate$_i$

因为,empty$_{i+1}$ < 2 × Realrate$_i$ × T

所以,full$_{i+1}$ > Length(Buffer)-2 × Realrate$_i$ × T-Wsize

full$_{i+1}$ > (2 × Max(Prerate$_i$, MaxComputeRate) + MaxComputeRate) × T-2 × Realrate$_i$ × T

因为,Prerate$_{i+1}$ > MaxComputeRate

所以,full$_{i+1}$ > 2 × Prerate$_i$ × T-2 × Realrate$_i$ × T + MaxComputeRate × T

所以,full$_{i+1}$ > MaxComputeRate × T

情况 2　假设 Realrate$_i$ > Prerate$_i$

在 $T_i \cdots T_{i+1}$时段产生的流数据流入关键时点 T_i 定义的"流入区",由于实际流速大于预测流速.

所以 $T_{i+1} = T_i + T'$($T' < T$),在 T_{i+1}时点,占满"流入区",Prerate$_i$ × T = Realrate$_i$ × T',在关键时点 T_{i+1},将 T_i关键时点定义的"流入区"划归到"处理区".

因为 Prerate$_i$ > MaxComputeRate,

且 full$_i$ ≥ MaxComputeRate × T,

所以,在关键时点 T_{i+1},full$_{i+1}$ = Realrate$_i$ × T' + MaxComputeRate × (T-T')(T' = Prerate$_i$ × T/Realrate$_i$),所以 full$_{i+1}$ > MaxComputeRate × T.

由于 empty$_i$ < 2 × Realrate$_i$ × T,所以需要增加空间使 empty$_i$ 超过标准值. 需要增加的空间数为 AddSpace = 2 × Realrate × T-empty. 证毕.

随着数据流的持续流入,系统执行算法 1,监测数据流流速,当数据流速异常加快时,在满足实时度 T 约束的情况下,算法 1 通过调用算法 2 和算法 3,采取积极降载与增加空间策略,实现数据流处理率的最大化.

算法 1　数据流监测算法

```
输入参数:Rcsw,Rb
输出结果:更新后的 Rb
While(TRUE)
{系统监测是否流入流数据;
    计时,并获得 T';
    if(T' > = T,或者"流入区"空间被占满)
    { 产生关键时点 Ti = T';
      计算 RealComputeRate(i-1),Realrate(i-1);
        if (RealComputeRate(i-1) > = MaxComputeRate)
        {更新 MaxComputeRate;}
      Prerate(i) = Realrate(i-1);
      PreComputeRate(i) = RealComputeRate(i-1);
      预估"流入区"空间;
      max(Prerate(i), MaxComputeRate) × T
      if (Prerate(i) > = MaxComputeRate)
      {计算空闲空间标准值;
        if (empty < 空闲空间标准值)
        { count = 空闲空间标准值-empty;
          AddSpace (Rcsw, Rb, count) ; }}}
}
```

算法 2　环形缓冲区降载 UpLoad(Rcsw,Rb)

```
输入参数:Rcsw,Rb
输出结果:更新后的 Rcsw,Rb
  if (环形缓冲区不处于降载状态){
```

if (“处理区”的长度超过 MaxComputeRate × T){
计算“待处理区”、“忽略区”空间数;
计算 Rcsw 的新的流数据入口位置;
重新计算 Full;}}

算法 3 环形缓冲区增加空间 AddSpace (Rcsw, Rb, count)

输入参数:Rcsw, Rb, count
输出结果:更新后的 Rb
UpLoad(Rcsw, Rb);
获取 Rb 中的空闲空间;
if (Rb 中的空闲空间不够)
{计算需增加的空间数;
if (系统内存空间足够)
{增加空间 count;}}

3.1 满足实时度要求的数据流异常减速检测算法

性质 3 当数据流速减速,并最终保持稳定,且 Realrate < MaxComputeRate,则,环形缓冲区的空间数也将稳定在最小值上,即,3 × MaxComputeRate × T + Wsize 时,此时“流入区”的长度保持为 MaxComputeRate × T,“处理区”的长度等于 RealRate × T,所有流入环形缓冲区的流数据均被处理,即,处理率 P 等于 100%.

下面通过具体的例子来说明环形缓冲区的变化.假设,Realrate_{i-2} > MaxComputeRate > Realrate_{i-1} = Realrate_i.

在关键时点 T_{i-1},“处理区”存储单元数为 $\text{Realrate}_{i-2} \times T$,环形缓冲区满足空闲存储单元标准值要求,环形缓冲区空间总数 $3 \times \text{Realrate}_{i-2} \times T$ + Wsize,系统预估 $T_{i-1} \cdots T_i$ 时段流速为 Realrate_{i-2},预估“空闲区”与“流入区”存储单元数均为 $\text{Realrate}_{i-2} \times T$,此时如图 1 所示.

在关键时点 T_i,“处理区”存储单元数为 $\text{Realrate}_{i-1} \times T$,环形缓冲区满足空闲存储单元标准值要求.预测 $T_i \cdots T_{i+1}$时段流速为 Realrate_{i-1}(MaxComputeRate > Realrate_{i-1}),预估“空闲区”与“流入区”需要存储单元数均为 MaxComputeRate × T,此时空闲存储单元数充足,流速减速时,关键时点 T_i 状态示意图如图 4 所示.

在关键时点 T_{i+1},“处理区”存储单元数为 $\text{Realrate}_i \times T$($\text{Realrate}_{i-1}$ = Realrate_i),预测 $T_{i+1} \cdots T_{i+2}$时段流速为 Realrate_i(MaxComputeRate > Realrate_i),预估“空闲区”与“流入区”需要存储单元数均为 MaxComputeRate × T,此时环形缓冲区满足空闲存储单元标准值要求,流速减速时,关键时点 T_{i+1}状态示意图如图 5 所示.减少环形缓冲区空间总数,直到环形缓冲区空间总数为最小值 3 × MaxComputeRate × T + Wsize,减少空间后,关键时点 T_{i+1}状态示意图如图 6 所示.

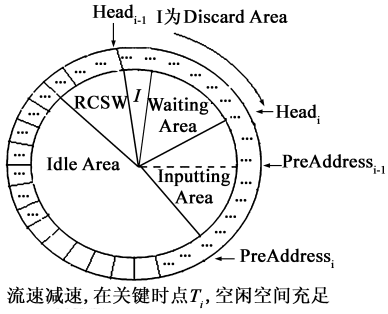

图4 流速减速时,关键时点T_i状态示意图

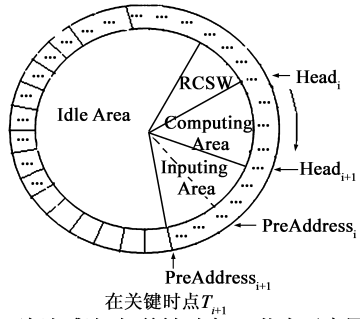

图5 流速减速时,关键时点T_{i+1}状态示意图

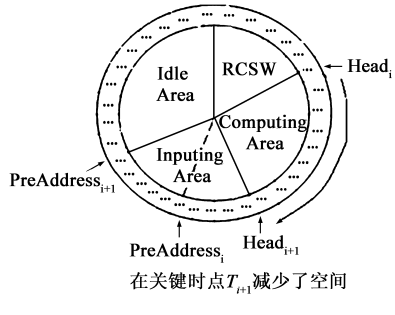

图6 减少空间后,关键时点T_{i+1}状态示意图

算法 4 减速检测与应对策略算法

输入参数:Rcsw, Rb
输出结果:更新后的 Rb
While(TRUE)
{ 系统监测流数据实际流速;
计时,并获得 T';
if($T' >= T$)
{
if(RealRate < MaxComputeRate,且环形缓冲区的存储单元数大于最小环形缓冲区 MinBuffer 的存储单元数){
计算空闲空间标准值;
if (空闲存储单元 > 空闲空间标准值,且不处于降载状态){
定义可以减少的空闲存储单元数;
减少环形缓冲区的存储空间;}}

系统监测到数据流的流速持续减慢,如果环形缓冲区的存储单元数大于最小环形缓冲区的存储单元数,执行算法 4,算法 4 采取惰性策略,继续监控,当实际数据流的流速小于最大处理速度时,此时如果空闲存储单元大于空闲空间标准值,则开始减少空间,采取惰性减少空间与积极降载的策略,可以减少空间容量的频繁变化,提高数据流挖掘效率,实现用宽裕的空间来换取数据流挖掘效率的提高.

4 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震荡数据流.实验

中,数据流持续流入缓冲区,系统监测与数据流处理同步运行.数据流处理环节被简化为计算流数据平均值.所有实验取值是在相应条件下所有实验值的平均值.取流数据量为 2000,实时度 T 为 0.5s,初始缓冲区大小为 75.缓冲区的存储空间随着流数据处理速度的加快而迅速增加,由于算法采用惰性减少空间的策略,所以流速振荡减速时,缓冲区的存储空间不会迅速减少,使得算法能够高效的应对流速无规律振荡,持续实施流数据的抽取.因此初始缓冲区的大小对实验结果没有明显影响.

实验 1　在流速无规律振荡情况下,考查数据流处理时间与处理率.流速无规律振荡情况下实验结果如图 7 所示.其中,系列 A、系列 B、系列 C 的流速无规律振荡范围分别为每秒 420 流量至每秒 150 流量之间、每秒 390 流量至每秒 220 流量之间、每秒 420 流量至每秒 300 流量之间.系列 A、系列 B、系列 C 的平均处理时间分别为 8524.865ms、6705.922ms、5684.815ms.实际平均处理率是系统运行过程中统计的实际处理流数据个数与流数据总量之比.估算平均处理率是平均处理速度与平均流速之比.观察图 7,系列 A 的平均流速和平均处理速度分别为 240.59 和 217.44,因此估算平均处理率为 0.903,非常接近实际平均处理率 P,系列 A 实际处理了 92.5%的流数据,由此,在系列 A,可以计算出每一单位流数据处理时间约为 4.6ms,同理,在系列 B 和系列 C,计算出每一单位流数据处理时间约为 4.7ms.所以,在不同振荡系列中,每一单位流数据处理时间相对稳定.由于系列 C 平均流速达到了每秒 399.23 流量,而系统的流数据处理能力远远低于平均流速,因此比较多的数据流无法满足实时度 T 的要求而被丢弃,导致实际平均处理率 P 仅为 0.603,它所处理的数据流数量低于其它系列,系列 C 的平均处理时间最短.实验一表明,在流速无规律振荡情况下,异常检测算法是稳定的,能够实时监控数据流的流速变化,实现对流数据处理率的智能调整,满足系统对实时度 T 的要求.

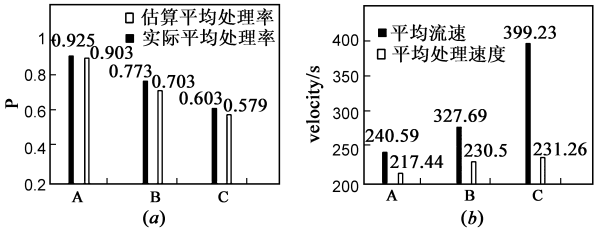

图7　流速无规律振荡情况下实验分析

实验 2　将数据流速控制在每秒 150 流量至每秒 420 流量之间.在流速递增或递减情况下,考查数据流处理时间与处理率.

在数据流流速递增和递减的情况下,实际平均数据流处理率 P 分别约等于 0.918 和 0.906,平均数据流处理速度分别约等于每秒 212.11 流量和每秒 212.29 流量,平均数据流处理时间分别约等于 8656.62ms 和 8541.59ms.从以上数据可以看出,在数据流流速递增和递减的情况下,平均数据流处理率 P 和平均数据流处理速度非常接近,实验 2 表明异常检测算法是可靠的,能够实时监控数据流的流速变化,实现对数据流的智能抽取.

实验 3　在数据流速不同的情况下,考查实时度 T 变化对数据流处理时间与处理率的影响.

实验结果如图 8 所示.分析比较平均数据流速分别为每秒 230 流量、每秒 300 流量、每秒 500 流量时实时度 T 的变化对数据流处理时间与处理率的影响.当平均数据流速为每秒 230 流量时,实时度 T 的变化不影响流数据处理率,P 均达到 1,在实时度 T 为 0.05s 时,总体数据流处理时间比实时度 T 为 0.5 和 1 时相对较短.当平均数据流速为每秒 300 到 500 流量时,流数据处理率 P 与数据流处理时间随着实时度 T 的增大而增大.

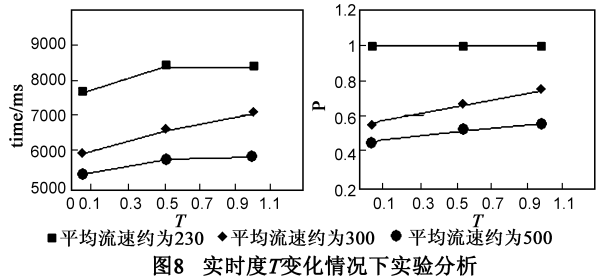

图8　实时度T变化情况下实验分析

实验结果表明,在数据流速小于等于数据流最大处理速度时,较小的实时度 T 能够使流数据更及时地被处理,在数据流速大于数据流最大处理速度时,实时度 T 的增大可以提高流数据实际处理率.并趋近于平均处理时间与平均流速之比.

5　结束语

由于许多应用领域产生数据流的流速不断地震荡,使得面向数据流的挖掘变得困难.本文探讨了基于 RCSW 的数据流速度异常检测算法,监测数据流速,预测流速,在监控到数据流速异常减速或增速时,系统通过调节环形缓冲区的空间大小来应对异常,为解决数据流处理能力与流速、流量与有限空间之间的矛盾提供解决方案.实验表明此算法能够智能应对流速度不可估,振荡变化的数据流挖掘环境,算法能够在流速突增的情况下,迅速增大环形缓冲区存储空间;在流速持续变慢的情况下,环形缓冲区减少存储空间;并满足流数据处理的实时度要求.本研究为数据流挖掘提供了一个面向 震荡数据流的可靠的数据流抽取算法.

参考文献

[1] VITTER J S. Random sampling with a reservoir[J]. ACM Trans on Mathematical Software, 1985, 11(1): 37 – 57.

[2] GIBBONS P B, MATIAS Y. New sampling-based summary statistics for improving approximate query answers[A]. Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data[C]. New York: ACM, 1998. 331 – 342.

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

[4] 李国徽,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-Wei, 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] 詹英,吴春明,王宝军.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[J].电子学报,2011,39(4):2262 – 2267.
ZHAN Ying, WU Chun-ming, WANG Bao-jun. An algorithm for data stream sampling based on ring circular sliding window tightly-coupled with buffer[J]. Acta Electronica Sinica, 2011, 38(2): 321 – 326. (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 al. 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-guang, 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] Lee L, Ting H. A simpler and more efficient deterministic scheme for finding frequent items over sliding windows[A]. Proceedings of the Twenty-Fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems[C]. ACM Press, 2006. 290 – 297.

[13] 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.
QIAN 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