滑动窗口连续查询结果存储优化

唐向红 李国徽
(华中科技大学计算机科学与技术学院 武汉 430074)

摘 要 在数据流滑动窗口查询研究领域中,考虑查询结果失效的连续查询成为了一个新的研究热点。查询结果的维护代价直接影响连续查询效率。根据对不同更新模式连续查询结果的分析,提出了一种带分支链表的梯队列来维护滑动窗口连续查询结果。它利用分支链表结构收集具有相同截止期的数据,采用梯队列的"产卵"机制,能适应具有各种不同分布的数据维护,且能达到O(1)的均摊(amortized)时间复杂度。实验表明,该结构显著提高了滑动窗口连续查询效率,明显优于同类结构。

关键词 梯队列,数据流,查询处理,存储优化
中图法分类号 TP311 **文献标识码** A

Storage Optimization for Continuous Query over Sliding Window Based on Ladder Queue

TANG Xiang-hong LI Guo-hui
(College of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract Query processing optimization based on update pattern awareness is a new hot topic in the research field of continuous queries over sliding window. Efficiency of continuous query processing highly depends on overhead of result state maintenance. This investigation proposed a ladder queue with branch lists to maintain result state of continuous query. The trunk list and the branch list were designed into the ladder queue. The ladder queue used the branch lists to gather the result tuples with the identical expiration time, and employed "spawning" mechanism to achieve O(1) amortized access time complexity for result data under the different distributions. Our experiments showed that the new ladder queue can improve the performance of query processing greatly and outperforms other data structures.

Keywords Ladder queue, Data stream, Query processing, Storage optimization

近几年来,随着信息技术的发展,出现了一类新的数据模型——数据流。它以实时、连续、有序的数据序列方式存在于人们生产和生活各个领域,如股票交易、火车售票系统、传感器网络等。它具有数据量大、连续快速、不可预测和短暂易逝等特点。数据流的这些特点决定了很多传统数据库查询处理技术无法推广到数据流上。它要求数据查询处理技术具有在线处理能力,能在有限的空间里实时地处理源源不断流入的数据并及时将处理结果反馈给用户。在很多数据流实际应用中,人们往往只关注数据流中离当前比较近的数据,对较远的历史数据兴趣不大。为了满足这种应用需要,滑动窗口技术孕育而生。滑动窗口(包括基于时间的和基于数量的)内的数据是数据流离当前最近的一段数据。滑动窗口内的查询处理是当前数据流查询的一个重要领域。

1 相关工作

近年来,国内外很多研究人员致力于提高滑动窗口内连续查询的性能[1-6],并有一大批研究成果问世。连续查询的一个重要特征是它们的输入和输出都与时间相关。当有新数据到达时,查询有新的结果产生。当滑动窗口向前滑动时,很多旧的数据因过期从窗口内移出,先前查询结果中就有一部分失效了。如何及时准确地反映连续查询在当前滑动窗口内的结果是连续查询面临的一个较大的挑战,它直接影响着数据流连续查询的效率。虽然国内外对连续查询研究较多,但是鲜有关注查询结果维护。

早期的研究只是简单地将数据流上的查询分为单调和非单调的。这里的单调指的是查询结果永不失效。而且认为只有单调的查询在无限数据流上是可行的[5,6]。由于认为查询结果永不失效,因此没有查询结果需要删除。

文献[7]虽然认为大部分连续查询是非单调的,不过他们仅仅认为查询结果失效次序是先进先出的,没有将非单调查询进一步分类。因此,查询结果可以使用普通的先进先出的队列来维护。

文献[4]和文献[8]采用直接方法和消极元组(negative tuple)方法来维护查询结果的正确性。直接的方法就是在每个查询结果的元组加上一个截止期属性,截止期一到就直接从结果中删除。消极元组方法就是当滑动窗口内有元组过期

到稿日期:2009-07-07 返修日期:2009-10-11 本文受国家高技术研究发展计划(863计划)项目(2007AA01Z309),国家自然科学基金(60873030)和国家国防预研基金(9140A04010209JW0504和9140A15040208JW0501),湖北省自然科学基金资助。
唐向红(1979—),男,博士生,主要研究方向为实时数据库系统、数据挖掘等,E-mail:txhwuhan@163.com;李国徽(1973—),男,博士,教授,博士生导师,主要研究方向为现代数据库工程、实时数据库系统等。

时，由滑动窗口产生一个与过期元组对应的消极元组，通过查询计划(query plan)来消除过期元组对查询结果的影响[4]。直接的方法在多滑动窗口上执行起来比较慢。而消极元组方法对一个元组需要处理两次，所以代价较大。

文献[9]首次将连续查询结果的产生和删除在时间上的次序定义为连续查询的更新模式。将滑动窗口上的连续查询更新模式细分为4类：单调、最弱非单调、弱非单调、严格非单调。单调指的是查询结果永不过期。最弱非单调指的是所有查询结果的生命期都是确定的，而且都等于滑动窗口的长度，即是先产生的结果先过期，这类查询有 selection 和 projection 等。弱非单调指的是查询结果的生命期是确定的但不等长，这类查询操作符有 join 和 distinct。严格非单调指的是查询结果产生后生命期不确定，何时过期与窗口未来的输入有关，两个滑动窗口之间 negation 就属于此类。文献[10]采用类似日历队列(Calendar Queue)结构来维护最弱非单调、弱非单调、严格非单调的查询结果。而日历队列有下列缺陷[11]：

(1)不能适合各种不同的分布，在各种分布情况下性能差异很大。对有很大偏斜的数据分布性能最差。

(2)它调整时期间隔的机制不够合理，会造成队列的抖动，调整次数多。因为调整付出的时间和空间代价都很大，所以效率很低。

(3)排序的时期不对，会增加算法的时间复杂度。

(4)抽样的算法太简单，不适合复杂的数据分布情况。

这种数据流连续查询结果的维护与离散事件模拟(discrete event simulation)研究领域中的未决事件集(pending event set)的维护很类似。未决事件集的维护过程就是每次都删除最小优先级的事件，而连续查询维护中总是删除过期的查询结果，这些过期的结果的截止期恰恰是查询结果中截止期最小的。

用于维护未决事件集的优先队列算法有很多种，主要分为基于多列表的结构和基于树的结构。基于列表的结构，代表有日历队列(calendar queue)[10]和懒惰队列(lazy queue)[12]。基于树的结构，主要代表有伸展树(Splay Tree)[13]和偏斜堆(Skew Heap)[15]。其中日历队列和懒惰队列能达到的O(1)的均摊访问时间复杂度，但是它们对数据分布规律比较敏感，对某些分布效率很低。伸展树和偏斜堆具有 $O(\log_2 N)$ 的均摊访问时间复杂度，其中 N 表示队列的元素个数。它们对数据的分布不敏感，具有比较稳定的性能。

最近出现的梯队列(Ladder Queue)[11]对各种数据分布都具有O(1)均摊访问时间复杂度。但是当出现较多相同优先级事件时，它会产生多级甚至无限级梯结构。

本文的工作主要包括以下几个方面：在分析滑动窗口的查询结果分布的基础上，将主干链和分支链结构嵌入到梯队列，它利用分支链表结构收集具有相同截止期的数据，克服了传统梯队列的缺陷；采用梯队列的“产卵”机制，能适应具有各种不同分布的查询结果，且能达到O(1)的均摊(amortized)时间复杂度。理论和实验表明，本结构显著提高了滑动窗口连续查询效率，效果明显优于同类结构。

2 相关背景说明及问题描述

图1说明了本文使用的滑动窗口模型 $W(S,\omega,\beta)$，其中 S 是滑动窗口基于的数据流，ω 是滑动窗口的宽度，β 是滑动窗口滑动的步长。当有元组进入滑动窗口时，它首先被缓存在缓冲区 Buffer 里。等到滑动窗口开始滑动时，将缓冲区里的元组加上统一的时间戳，放入滑动窗口内。

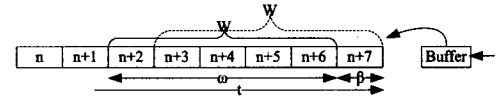

图1 滑动窗口模型

从上面的模型可知，进入滑动窗口的很多元组具有相同的时间戳。因此，查询结果中的很多元组就有可能具有相同的截止期。

本文将所研究的滑动窗口连续查询存储优化的问题描述为动态维护运行在一个或者多个数据流滑动窗口上非单调的连续查询的结果，保证其结果的有效性。同时能在各种数据分布情况下，数据维护的均摊(amortized)时间复杂度[15]为O(1)。

3 带分支的梯队列构造及操作

带分支链表梯队列跟普通梯队列一样分为三层结构(如图2所示)，顶层(Top)是由一个简单的无序的链表构成；中间(Ladder)是梯层结构，它由几层像楼梯一样的结构构成，每个梯结构中由多个桶构成，每个桶内包含着一个无序的链表；底层(Bottom)是由一个有序的链表构成。带分支的梯队列与普通的梯队列的区别主要有3个方面：

(1)链表结构不同。普通梯队列中的链表只是普通的单向链表结构，而带分支的梯队列中的链表由主干链表和分支链表构成。主干链表是一个双向链表，在主干链表中的每个主干结点包含一个查询结果的元组外，还有一个头指针和尾指针，它们分别指向主干结点所带的分支链表的头和尾。分支链表是一个单向链表，分支链表中的分支结点内包含具有相同截止期的元组。

(2)元组映射方式不一样。普通梯队列中，一个元组对应于队列中的一个结点。而在带分支的梯队列中，一个元组可能对应队列中一个主干结点，也可能对应于一个分支结点。

(3)元组离开队列的方式不一样。在普通梯队列中，每次出列通常是一个元组。而在带分支的梯队列中，可能一次出列多个元组。

图2中展示的是带分支链表梯队列基本结构。图中T代表一个主干结点，b代表一个分支结点，O表示一个被清空的桶，C表示该桶中的结点正在被排序准备转移到底层队列中，U表示该桶中的结点还没有被排序。表1是元组入出队列需要用到的参数。

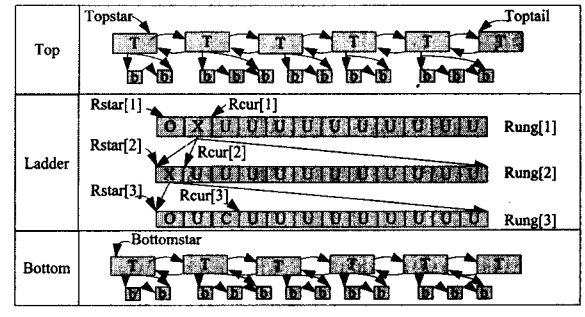

图2 基本的带分支的梯队列

表 1　各层用到的参数及含义

层次	变量参数名	定义及含义
Top	MaxET	表示顶层链表中主干结点元组的最大截止期
	MinET	表示顶层链表中主干结点元组的最小截止期
	TopST	表示进入顶层队列所需阈值，只有大于它才能进入顶层队列
Ladder	BkWd[x]	表示第 x 梯层的桶的宽度
	NB[i,j]	表示第 i 层的第 j 桶中的主干结点的个数
	NR	表示共有多少梯层
	RC[x]	第 x 层中当前正在排序转移的桶的最小截止期
	RS[x]	第 x 层的最小截止期
	THRES	表示一个阈值，如果一个桶或者底层链表的主干结点数超过它，则要产生一个新的梯结构
Bottom	NBot	最底层链表中的主干结点的个数

3.1　入队列操作

元组进入带分支的梯队列分为两种情况：

(1)队列为初始状态，即顶层、中间层和底层都为空时，若有元组需要进入队列，这个元组将作为一个主干结点直接插入顶层的主干链表中，同时将该元组的截止期赋给 $MaxET$ 和 $MinET$。

(2)当梯队列中不为空时，一个元组 TP 进入队列，设它的截止期为 ET。如果 $ET \geqslant TopST$，元组 TP 将要插入到顶层链表中。如果顶层链表为空，则作为一个主干结点插入到主干链表中。否则，从顶层的主干链表尾部开始，向前查找 α 个结点。如果能找到与元组 TP 具有相同截止期的主干结点，则将元组 TP 插入该主干结点所带的分支链表的尾部。否则，元组 TP 将作为一个主干结点插入到主干链表的尾部，同时更新 $MaxET$ 和 $MinET$ 的值。如果 $ET < TopST$，元组 TP 将要插入到中间层或者底层队列中。接着将 ET 与 $RC[1], RC[2], \cdots, RC[NR]$ 进行比较。一旦 $ET \geqslant RC[i]$，就能确定元组 TP 将插入第 i 层梯结构。再根据式(1)计算出具体桶的序号 j。如果该桶的主干链表为空，则作为一个主干结点插入到主干链表中，$NB[i,j]$ 自增 1。否则，从该桶的主干链表尾部开始，向前查找 α 个结点。如果能找到与元组 TP 具有相同截止期的主干结点，则将元组 TP 插入该主干结点所带的分支链表的尾部。如果不能找到与元组 TP 具有相同截止期的主干结点，元组 TP 将作为一个主干结点插入到该桶主干链表的尾部，同时 $NB[i,j]$ 自增 1。如果元组 TP 不能插入顶层和中间层，则它只能插入有序的底层链表。从底层主干的链表尾起，如果能查找到与元组 TP 具有相同截止期的主干结点，则元组 TP 作为一个分支结点插入该主干结点所带的分支链表的尾部。否则，元组 TP 将作为一个主干结点插入底层主干链表并保持底层主干结点链表的有序性。α 的取值与结果的截止期分布有关，通常情况下 α 取值 3。

3.2　出队列操作

当结果中的元组截止期即将到达，则需要将过期的元组从队列中删除出去。这个过程分为 3 种情况：

(1)中间梯层和底层都为空时，将顶层中的主干结点转移到中间的第一层结构 $Rung[1]$ 中。第一层结构中每个桶的宽度 $BkWd[1]=MinPS$。$MinPS$ 为参与查询的滑动窗口里的最少步长，即 $MinPS=\mathrm{Min}\{\beta_1, \beta_2, \cdots, \beta_{NumSW}\}$。这样做的好处是使得具有不同截止期的元组尽量分布到不同的桶中，方便具有相同截止期的元组尽可能聚合到分支链表里，减少整个队列中主干结点的个数，降低出列代价。将 $MinET$ 值赋于 $RS[1]$ 和 $RC[1]$。将顶层的 $TopST$ 的值设置为 $MaxET + BkWd[1]$。设任意一个顶层链表中主干结点 tTP 的截止期为 $ExTP$。按照式(1)将主干结点 tTP 分配到第一层梯结构中相应的桶中，桶里包含这个带有分支的主干链表。

$$Bucket\_index=\left\lfloor \frac{ExTP-RS[i]}{BkWd[i]} \right\rfloor \tag{1}$$

式中，$Bucket\_index$ 表示该元组应该放入的桶的索引号。将元组放入第 $Bucket\_index$ 桶的过程，跟入队操作差不多。从桶里的主干链表末尾起搜索 α 个主干结点，如果能找到一个跟 tTP 具有相同截止期的主干结点 $btTP$，则 tTP 以及它所带的分支结点都将作为 $btTP$ 的分支结点插入到 $btTP$ 的分支链表尾端。否则，tTP 将作为一个主干结点插入到桶里的主干链表末尾，同时 $NB[1, Bucket\_index]$ 增加 1。

一旦顶层中的所有元组都转移到第一层梯形结构中，接着从第一个桶开始遍历第一层梯结构，寻找第一个非空的桶，即它里面至少包含一个结点。每跳过一个空桶，则 $RC[1]$ 自增 $BkWd[1]$。

假设找到一个非空的桶 B_c，它包含的主干结点个数为 CB_c。如果 CB_c 不超过 $THRES$，则将 B_c 中的主干结点采用稳定排序方法按截止期升序排列，再将链表移到底层(Bottom)结构中。从底层(Bottom)链表头开始出列所有截止期小于当前时间的主干结点和它们所带的分支结点。

如果 CB_c 超过 $THRES$，“产卵”机制就会启动，即会产生下一层梯结构 $Rung[2]$。新产生的梯层结构中，桶的宽度由式(2)计算产生：

$$BkWd[i]=\frac{BkWd[i-1]}{THERS} \tag{2}$$

式中，$i \geqslant 2$。再根据式(1)，将 B_c 里所有元组分配到 $Rung[2]$ 中，过程跟将顶层的元组分配到 $Rung[1]$ 的过程一样。重复上述过程，直到找到一个第 t 层梯结构 $Rung[t]$ 的非空且主干结点个数不超过 $THRES$ 的桶，将其主干链表按截止期升序排序，放入底层(Bottom)结构中。从底层(Bottom)链表头开始出列所有截止期小于当前时间的主干结点和它们所带的分支结点。

(2)底层为空而中间梯层不为空。在最低的梯结构 $Rung[n]$ 中，从 $RC[n]$ 开始查找非空桶。每跳过一个空桶，则 $RC[n]$ 自增 $BkWd[n]$。后面的处理步骤跟上面第(1)种情况提到的过程一样，不再重述。当某一层梯结构中所有元组都出列完了，将该层梯结构删除。

(3)底层不为空时，从底层(Bottom)链表头开始出列所有截止期小于当前时间的主干结点和它们所带的分支结点。

3.3　*THRES* 取值说明

当进入队列的元组截止期出现偏斜分布时，这些元组进入第一层梯结构，某些桶里会出现很多元组，而有些桶几乎没有元组。$THRES$ 决定着何时产生下一级梯结构。产生下层梯结构的主要目的是希望元组均匀分布到下层梯结构的桶中，在转移到底层排序时提高效率。产生下层梯结构还可以进一步聚合具有相同截止期的元组，从而减少主干结点数目，提高效率。

由第一层梯结构的桶宽取值可知，在每个桶里，来自同一个滑动窗口的元组具有相同的截止期，在同一个桶里最多具有 $NumSW$ 不同的截止期，$NumSW$ 是连续查询所涉及的滑动窗口数。当一个桶里的主干结点个数超过 $NumSW$，说明具有相同截止期的元组还没有全部聚集到一个主干结点上

来。同时考虑当排序元素的个数不超过50，线性排序具有很高的效率[16]。因此，当桶中主干结点个数超过 Min{50，*NumSW*}，即 *THRES* 取值为 Min{50，*NumSW*}时，产生下级梯结构，具有较高的效率。

4 带分支结构梯队列的优势

使用带分支结构梯队列用于维护滑动窗口的连续查询结果，有以下优势：

(1)能适合各种不同的分布情况。当查询结果的截止期呈均匀分布时，带分支结构梯队列的中间梯结构就只有一层不会产生新的梯结构，跟普通的日历队列一样，具有较高的效率。当查询结果的截止期呈偏斜分布时，中间的梯结构部分就会产生新的较低级的梯结构，将聚集较多元组的桶进一步细分，使其均匀分布到下一级梯结构中，这样比普通的日历队列更加有效率。

(2)很适合维护滑动窗口的连续查询结果。从前面的分析可以知道，在连续查询结果中，会出现多个数据具有相同的截止期。带分支结构梯队列通过分支结构，将大多数具有相同截止期的数据聚集在同一个分支链表里，这样既保证查询结果的稳定性，又能保证查询结果的实时性。

(3)克服了普通梯队列产生无限级梯结构的缺陷。在通常的梯队列中，如果进入队列的多个事件具有相同的优先级，就可能产生无限级梯结构。而带分支结构的梯队列，彻底克服了这个缺陷，具有相同优先级或者截止期的数据随着从顶层到中间层过程中不断聚集到分支链表里，在带分支结构的梯队列里，每个主干结点及它所带的分支链表在队列里被当作一个元素对待，因此带分支结构梯队列从根本上消除了产生无限级梯结构的可能性。

(4)减低了出入队列的时间。在带分支结构的梯队列中，每个主干结点及它所带的分支链表都被当作一个元素来处理，从顶层到中间多级梯结构之间元组转移过程中，大大减低了处理的时间复杂度。

5 性能分析及实验结果

5.1 理论分析

性质1 带分支结构的梯队列具有不超过 O(1)的均摊时间复杂度。

证明：设带分支链表的梯队列有 N 个元组，在这 N 个元组中假设平均 p 个元组具有相同的截止期，在入出队列过程中这 p 个元组可能生成 q 个主干节点($1\leqslant q\leqslant p$)，将主干结点及其所带的分支结点当成一个队列里一个元素看待，则 N 个元组分支结构的梯队列就成了一个含有 $\frac{Nq}{p}$ 元素的通常的梯队列。由于通常的梯队列具有 O(1)的均摊时间复杂度[11]，而且 $\frac{Nq}{p}$ 小于 N，因此 N 个元组带分支结构的梯队列均摊时间复杂度不超过 O(1)。

5.2 实验结果分析

本节对所提出带分支链表的梯队列进行性能测试，选取日历队列(calendar queue)、普通梯队列(ladder queue)为对照数据结构。日历队列是文献[9]用来存放查询结构的数据结构，普通梯队列是带分支结构梯队列改进的基础，都与本文研究问题相关。因此我们选取它们作为对比结构，具有一定的可比性。将这3种数据结构都嵌入到更新模式觉醒的查询处理器中。实验平台配置如下：Intel™ Core 2 Duo 1.4GHz / 1G，Windows XP(professional 版)，所有代码都用 VC++(6.0)实现。实验所使用的数据是美国劳伦斯国家实验室跟世界各地广域网的 TCP 连接记录。每条记录包括系统分配的时间戳、session 持续时间、协议类型、负载大小、源 IP 地址和目的 IP 地址。根据目的 IP 地址将它分为几个逻辑数据流。实验用典型的弱非单调操作符 join 和 distinct 作为代表操作符。

在图3中，Query 1 中的 Link1 和 Link2 是两个数据流，它们是对源 IP 地址的连接。Query 2 是对源-目标 IP 对的相异查询。

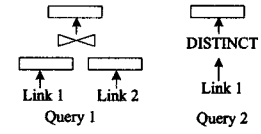

图3 join 与 distinct 的流查询操作

实验中，时间滑动窗口大小从 1s 变化到 1000s。在 Query 1 和 Query 2 中输入的数据流，数据流入速度从 1 个元组/s 到 100 个元组/s 不等。

实验性能评价指标主要是平均访问时间，它包括元组入队和出队的平均时间。

图4展示了三者在 5 元组/s 流入速度情况下的平均访问时间。随着窗口不断增大，join 的结果有所增大，日历队列的平均访问时间有一定幅度的上升，普通梯队列与带分支链表的梯队列的平均访问时间也有所增加。带分支链表的梯队列增幅最小。

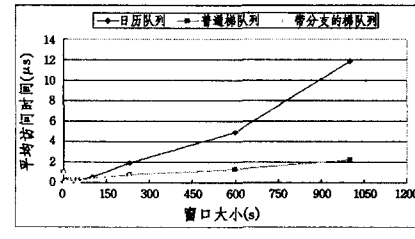

图4 Query 1 在流入速度 5 元组/s 的平均访问时间

图5展示的是 Query 1 在流速为 100 元组/s 情况下窗口较小时，3 种结构的性能很接近。在窗口增大过程中，由于流速较大，join 形成的结果就会急剧膨胀。在这种情况下，由于日历队列对数据截止期分布的敏感性，因此它的访问时间也大幅增长，远远高于其他两种结构。普通梯队列的性能也有一定程度的下降，这是由于 join 结果里的具有相同截止期的元组有所增加，这样会造成它的性能下降。带分支的梯队列，由于分支结构大大减少了主干结点的数目，因此性能受到的影响不大。

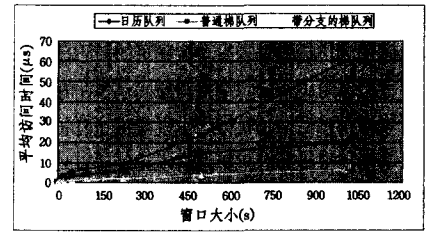

图5 Query 1 在流入速度 100 元组/s 的平均访问时间

图6和图7分别是 Query 2 在流速 5 元组/s 和 100 元

组/s 情况下的平均访问时间图。在这两个图中，随着窗口的增大，每种结构的访问时间都有所增大，带分支链表的梯队列具有最好的时间性能。图 6 和图 7 与图 4 和图 5 相比，访问增幅较小，这是因为当流速和窗口增大时，join 的增长速度比 distinct 差一倍左右，所以图 4 与图 5 的平均访问时间增长速度比图 6 和图 7 大得多。

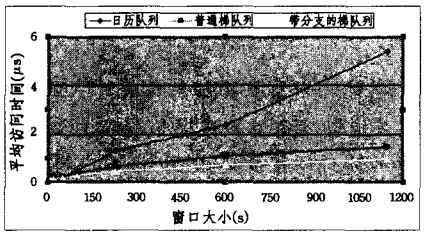

图 6　Query 2 在流入速度 5 元组/s 的平均访问时间

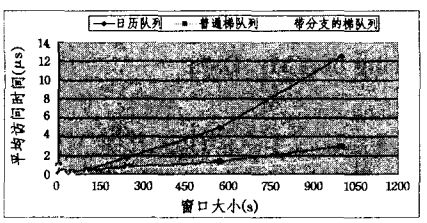

图 7　Query 2 在速度 100 元组/s 的平均访问时间

结束语　随着信息技术的不断发展，数据流在各个应用领域涌现，流入数据流查询越来越受到人们的关注。数据流上连续查询结果随着数据的不断流入流出也会不断变化。如果有效地维护数据流上连续查询的结果，很大程度上也就决定了连续查询的效率。当前鲜有研究涉及到数据流上连续查询结果的维护，特别是非单调连续查询结果的维护。

通过对连续查询结果的分析，本文设计了一种带分支链表的梯队列来维护连续查询结果。这种带分支链表的梯队列能适应具有各种不同截止期的数据的维护，而且能较大程度地降低数据平均访问时间。实验结果表明，带分支链表的梯队列能显著提高数据流的连续查询效率。

参 考 文 献

[1] Babcock B, Babu S, Datar M, et al. Models and Issues in Data Streams[C]// ACM Symposium on Principles of Database Systems(PODS). 2002,5:1-16

[2] Golab L, Ozsu M T. Issues in Data Stream Management[J]. SIGMOD Record 32,2003:5-14

[3] Golab L, Ozsu M T. Processing sliding window multijoins in continuous queries over data streams[C]// Proceedings of 29th International Conference on Very Large Data Bases(VLDB). 2003:500-511

[4] Hammad M, Aref W, Franklin M, et al. Efficient execution of sliding window queries over data streams[R]. Purdue University,2003

[5] Law Y N, Wang H, Zaniolo C. Query languages and data models for database sequences and data streams[C]// Proceedings of 30th International Conference on Very Large Data Bases (VLDB). 2004:492-503

[6] Terry D, Goldberg D, Nichols D, et al. Continuous queries over append-only databases[C]// Proceedings of ACM-SIGMOD 1992 International Conference on Management of Data(SIGMOD 1992). 1992:321-330

[7] Krämer J, Seeger B. A temporal foundation for continuous queries over data streams[C]// Proceedings of 11th International Conference on Management of Data(COMAD 2005). 2005:70-82

[8] Hammad M, Mokbel M, Ali M, et al. Nile: a query processing engine for data streams[C]// Proceedings of the 20th International Conference on Data Engineering (ICDE 2004). 2004:851

[9] Golab L, Ozsu M T. Update-Pattern-aware Modeling and Processing of Continuous Queries [C]// Proceedings of ACM-SIGMOD 2005 International Conference on Management of Data (SIGMOD 2005). 2005:658-669

[10] Brown R. Calendar queues: A fast O(1) priority queue implementation for the simulation event set problem[J]. Communications of the ACM,1988,31(10):1220-1227

[11] Tang W T, Goh R S M, Thng I L J. Ladder queue: An O(1) priority queue structure for large-scale discrete event simulation[J]. ACM Transactions on Modeling and Computer Simulation, 2005,15(3):175-204

[12] Onngren R, Riboe R J, Ayani R. Lazy queue: a new approach to implementing the pending event set[J]. International Journal in Computer Simulation,1993,3:303-332

[13] Sleator D D, Tarjan R E. Self-adjusting binary search trees[J]. Journal of the ACM,1985,32:652-686

[14] Sleator D D, Tarjan R E. Self-adjusting heaps[J]. SIAM Journal on Computing,1986,15:52-69

[15] TarjanRE. Amortizedcomputationalcomplexity[J]. SIAMJournal on Algebraic and Discrete Methods,1985,6:306-318

[16] Marín M. An Empirical Comparison of Priority Queue Algorithms[R]. Oxford University,1997

(上接第 105 页)

研究，包括网络存储系统可靠性评价指标的定义，在对典型网络存储系统分析的基础上，采用框图法理论，提出基于混联结构的网络存储系统可靠性分析方法，即一种与业务负载关联的系统可靠性计算方法。该方法相对于测试等其它方法具有简单、省时、费用低等特点，因此更有利于应用于网络存储系统的方案和设备优化。

参 考 文 献

[1] 郑纬民，舒继武. 下一代分布式智能网络存储系统的发展趋势[J]. 世界电信，2004，8：16-29

[2] 曾照洋，任占勇. 基于多状态的系统可靠性预计[J]. Aeronautic Standardization & Quality，2008，2：43-47

[3] 唐应辉. 计算机系统的可靠性研究[D]. 成都：电子科技大学，2006

[4] 姜宁康，时成阁. 网络存储导论[M]. 北京：清华大学出版社，2007：28-31

[5] 严南. 计算机应用系统中数据存储系统的可靠性研究[D]. 昆明：昆明理工大学，2006

[6] 杨照宏. 分布式海量存储系统的可靠性和容错性研究[D]. 大连：大连海事大学，2007

[7] 赖莉. 数据存储方案分析[J]. 渝西学院学报：自然科学版，2004，4：43-47

[8] Lynch M. The Storage Report——Customer Perspectives & Industry Evolution[R]. 2001

[9] Relex Software Co. Intellect. Reliability: A Practitioner's Guide [M]. 2005:145-151

[10] 金星，洪延姬. 系统可靠性与可用性分析方法[M]. 北京：国防工业出版社，2007：12-13