ETHs:n-of-N 模型下基于指数划分的一种数据流大纲维护算法

邱海艳 陈立军 赵加奎
(北京大学计算机科学与技术系 北京 100871)

摘 要 数据流大纲的维护对于 DSMS 非常重要:流数据的实时性、持续性和有序性(即,老化特性)使得查询引擎需要根据实时的概要信息自适应地调整执行计划,保持其执行效率。本文提出一种新的数据流大纲结构—ETHs,它通过指数划分方法将数据流在时间域上划分为指数区间,每个区间用具有较小空间复杂度和时间复杂度的 Tiny 直方图来记录区间的概要信息,使得 ETHs 既能够反映数据流上某些数据的衰减,又能够实现 n-of-N 模型下的共享计算,在 εN 误差范围内持续地维护最近 N 个元素的概要信息,具有较小的时间代价和空间代价。实验证明,ETHs 是数据流上的一种较理想的大纲结构。

关键词 数据流,大纲,n-of-N,等深,指数划分

ETHs: A Data Stream Synopsis Maintaining Algorithm Based on Exponential Partition in n-of-N Model

QIU Hai-Yan CHEN Li-Jun ZHAO Jia-Kui
(Department of Computer Science & Technology, Peking University, Beijing 100871)

Abstract Maintaining data stream synopsis is very important in DSMS. Data stream tuple is real-time, continuous and ordered (namely, aged). Query engine needs to adjust query plan by on-line synopsis to guarantee its processing efficiency. In this paper, we propose a new synopsis structure called ETHs, which partitions time dimension into exponential intervals by EH partitioning technique. In each subinterval, we use tiny histogram which has small space and time complexity to record summary information. So, it can reflect the stateness of certain data elements and share computations under n-of-N model. With a guaranteed precision of εN, it continuously maintaining the summary information of the most recent N elements over data stream with little time and space overhead. Performance study shows that ETHs is a good data stream synosis maintaining algorithm.

Keywords Data stream, Synopsis, n-of-N, Equi-Depth, Exponential partition

1 简介

随着对数据流管理系统(DSMS)的深入研究,我们需要解决很多新的技术难点,如何维护数据流上具有时间特性的统计信息即为其中之一。数据流具有时间特性,这主要是针对数据流上大量的窗口查询而言的。在传统数据库中,查询优化借助于对静态数据集的势的估算来制定查询计划。而在动态的数据流环境中,这是不可行的,数据分布的不均衡性使得某一时刻最优的执行计划在另一时刻可能不再最优。数据不是持久的,是瞬时的,对数据只能进行一遍处理,而窗口查询需要考虑对过期数据的处理,将所有历史数据保存在内存中是不可能的。我们需要通过算法来获得数据流的摘要信息,这种摘要信息必须简捷而且相当精确,这就是所谓的"大纲"。数据流查询引擎使用这些大纲信息来自适应地调整执行计划,保持其执行效率。但是传统的大纲是不足以满足应用需求的,因为大多数数据流应用中的数据会"老化",例如,最近一小时的热点查询可能并不是全天的热点查询,只是简单地累计历史数据会导致错误的结果。如果为每个查询都保持实时更新的大纲,则每当新元组到达或旧元组过期都需要更新所有查询的大纲,这势必会消耗大量系统资源,影响系统效率。因此我们需要寻找更适当的方法来描述历史数据。本文提出一个新的数据流大纲结构——ETHs,它将数据流划分为指数区间,每个区间用具有较小空间复杂度和时间复杂度的 Tiny 直方图来记录区间的概要信息,使得 ETHs 既能够反映数据流的指数式衰减,又能够从中获取任意大小区间的概要信息。采用这样的方法,我们能够在保证精确度的前提下有效地共享计算,虽然增大了计算时间,但是具有较小的更新时间代价和空间代价。

2 相关工作

数据流是一个元组序列,对数据流的处理有三个约束条件:单向性、实时性以及内存的限制。如果输入流的速度太快,以致查询处理器来不及处理;或注册到系统中的查询希望得到的查询结果能够包括已经"流过"的历史数据,就需要引入新的数据结构——大纲。将数据流以大纲的形式保存在内存中,数据流处理引擎根据大纲和流入的数据计算出查询的近似结果。我们用 N 来表示数据流中最近的元组,以此作为计算空间复杂度和时间复杂度的参数。理想地,我们期望内存的限制与 N 是无关的。但实际上,这是不可能的。如果能够在至多 $O(\mathrm{poly}(\log N))$ 空间和 $O(\mathrm{poly}(\log N))$ 时间内完成数据流大纲的计算,则所有问题都很好地解决了,但事实并非如此。我们只能试图寻找接近这个目标的算法。

通常的大纲处理算法一般包括采样、直方图和小波等。其中直方图是一种能够有效捕获数据分布的数据结构,它已得到广泛的应用。在数据流上也同样可以使用直方图来生成数据概要。但是以往对直方图维护算法的研究大多限于对所有历史数据的统计,如文[1~3],这些直方图不能体现数据的时间特性,因为越老的数据,其重要性越低。而将所有历史数

邱海艳 硕士生,主要研究方向为数据库和信息系统;陈立军 博士,副教授,主要研究方向为数据库与信息系统;赵加奎 博士生,主要研究方向为数据库与信息系统。

据都存储在内存中是不可行的。文[4]给出了第一个滑动窗口上近似直方图的算法，它是在精度和速度上权衡考虑的结果。但是若同时有大量滑动窗口出现，为每个滑动窗口都实时计算直方图，其代价是很高的。文[5]提出了一个称为指数直方图(exponential histogram，EH)的算法，它为多个滑动窗口维护一个近似度为 ε 的统计数据，用于回答 BasicCounting 问题。文[6]将 EH 的问题域扩展到多值值域，用两维混合直方图捕获数据分布的变化。但是这种直方图的构建和维护都很复杂，我们需要寻求能够有效生成数据流大纲同时又能够节省时间和空间的数据流直方图维护算法。本文的第 3 部分给出问题的定义，第 4 部分描述了具体的算法，并对算法进行了分析，第 5 部分对算法进行了试验，在最后我们给出结论。

3 问题定义

为简化描述，我们只考虑数据流关系中的一个属性 X，其值域为 D，我们用 T_c 表示当前的时间戳，若滑动窗口大小为 ω，则滑动窗口中任意数据元素在属性 X 上的域空间为 $D\times\{T_c-\omega+1,\cdots,T_c\}$，每个数据流元组都与一个时间戳 t_i 相关，我们用(x,t_i)表示滑动窗口中的数据流元组。每个时间单元，T_c 增加 1，滑动窗口向前滑动一步。这个过程实际上是对一些数据的插入和对一些数据的删除，插入的数据为(x,T_c)，删除的数据为$(x,T_c-\omega)$。显然，为每个滑动窗口都需要实时维护一个大纲。本文中的算法，以一个固定长度的队列为基础，随着流数据的到来依次构建 Tiny 直方图，所有 Tiny 直方图的元组总数有一个上界，即为 N。以指数划分的方法合并时间戳较老的 Tiny 直方图，滑动窗口的移动是以子区间的粒度为单位，而不是以单位时间或流元组为粒度，最终合成的概要数据实际上是对任何小于 N 的滑动窗口中元素频率分布的一种近似函数。

4 ETHs

4.1 n-of-N 模型

对于数据流上的大纲，有多种计算模型，一般来说，数据流上的滑动窗口可分为窗口大小固定和窗口大小可变两类，前者实际上是基于元组数目的滑动窗口，而后者是基于时间的窗口(最近 N 个时刻内的元组数目是不固定的)。对于数据流上大量存在的滑动窗口，我们需要记录它们的大纲信息，这样的代价是昂贵的。为了最有效地共享工作，我们可以采用 n-of-N 模型计算数据流大纲。n-of-N 模型可分为两类，一类是基于元组数目的，另一类则是基于时间的。在基于元组数目的 n-of-N 模型下，维护最近 N 个元素的大纲，从中能够得到最近 $n(\forall n\leqslant N)$ 个元素的大纲。在基于时间的 n-of-N 模型下，维护最近 N 个时刻的大纲，从中能够得到最近 $n(\forall n\leqslant N)$个时刻的大纲。在 n-of-N 模型下，无论是对于窗口大小固定还是窗口大小可变的滑动窗口，我们必须很快地从大纲中得到相关信息。本文中的算法是以基于元组数目的 n-of-N 模型为研究对象的，将其扩展到基于时间的 n-of-N 模型也是可行的。

4.2 指数划分技术

数据流是具有时间特性的，越老的数据，其重要性越低。因此，我们需要引入老化因子 λ 来体现这种时间特性[5]。若在时刻 t，流上的数据为：$X_1,X_2,\cdots,X_{t-1},X_t,\cdots$，则加入老化因子的流数据为：$\lambda^{t-1}X_1,\lambda^{t-2}X_2,\cdots,\lambda X_{t-1},X_t,\cdots$。这种指数式衰减的老化技术实际上是削弱了过时数据对当前状态的影响，因为对于大多数实时应用来讲，都不会对过期数据的概要感兴趣，在我们的 ETHs 中使用这种划分技术还有一个非常重要的原因，就是它对桶的数目是可约束的，EH(exponential histogram)在额定空间内能够更有效地体现数据的时间特性，这对于空间的有效使用是必须的。若相对误差为 ε，令 $\lambda=\lceil 1/\varepsilon\rceil/2+1$，我们称 λ 为指数划分的阈值(threshold)，在由 0 和 1 组成的简单数据流上，当新数据到来时，若新数据为 0，则等待下一个数据的到来；若新数据为 1，删除已过期的桶，创建一个大小为 1 的新桶，以该数据的时间戳作为新桶的时间戳，从当前桶开始往前遍历已存在的桶，如果大小为 2^{i-1} 的桶的个数超过 λ，则将最老的两个大小为的桶合并为一个大小为 2^i 的桶，以前者中最老的时间戳作为后者的时间戳，大小为 2^{i-1} 的桶的合并会导致大小为 2^i 的桶个数超过 λ，从而引起级联合并。在 EH 中，桶的个数至多为 $\left(\frac{k}{2}+1\right)\left(\log\left(\frac{2N}{k}+1\right)+1\right)$，每个桶所需的内存至多为 $\log N+\log\log N$ 位。每到来一个新元素，可在 O(1)时间，最坏情况下 O($\log N$)时间内完成处理。

4.3 Tiny 直方图

我们用指数划分技术将数据流上最近 N 个元素划分为一系列子区间，在每个子区间中构建 Tiny 直方图。这里，我们假设属性 X 的值集合为 V，V 是 D 的子集。令 $V=\{v_i:1\leqslant i\leqslant D\}$，其中，$i<j$ 时有 $v_i<v_j$。我们用 f_i 表示 v_i 的频率，则 X 的数据分布为 $T=\{(v_1,f_1),(v_2,f_2),\cdots,(v_D,f_D)\}$。属性 X 上的直方图 H 是根据某种划分规则将 T 划分到 β 个互不相交的子集中，也即所谓的桶[7]。不失一般性，我们假设每个桶中元素的属性值是“均匀分布”的，通过取桶频率的平均值来近似每个元素的频率[8]。这里，若 $b_{i-1}=v_n$，$b_i=v_m$，我们用 $\Delta_i=m-n$ 表示桶 i 中不同元素的数目，于是 H 可以表示为$\{(b_1,\Delta_1,f_1),(b_2,\Delta_2,f_2),\cdots,(b_\beta,\Delta_\beta,f_\beta)\}$。

本文中的 Tiny 直方图，其划分方式是等深的。我们需要计算 β 个桶的边界：$q_1,q_2,\cdots,q_\beta$，在统计学上，q_i 又称为分位数(quantile)，分位数是 T 中位于 i/β 的数据元素。我们需要找到这样的数据元素来作为桶的边界点。计算分位数的一个简单方法[9]是对关系 R 中的所有元组在属性 X 上以升序进行排序，并从中选择 $\beta+1$(包括第一个和最后一个)个位置，使得任意相邻的两个位置之间的元组数目相同。显然，这些位置为 $1,1+k,1+2k,\cdots,1+(\beta-1)*k,1+\beta^*k=N$，其中 $k=(N-1)/\beta$。我们选择位于这些位置的元组的 X 属性值作为桶的边界，从而完成等深直方图的构造。从这个直方图中，我们可以很容易地得到属性 X 的选择度估算值，与实际值相比，若 X 位于两个分位数之间，则其最大误差为 $1/2\beta$，若 X 位于分位数上，则其最大误差为 $1/\beta$。显然，若想减少误差，只需增大桶的数目。然而，桶的数目增多，占用的内存也会增加，因此，需要折中确定值。

4.4 ETHs

我们在前面讨论了 EH 直方图，这种直方图面向的是简单的 0、1 数据流，要想将其扩展到整个实数域上，还有很多问题。如何动态地根据数据集的改变对数值维和时间维进行合理的划分，是我们面临的挑战。在时间维上我们可以借鉴 EH 的划分方法，将时间空间划分为一系列指数子区间，滑动窗口的移动以子区间为单位。在每个子区间中构造 Tiny 直方图，在 Tiny 直方图中以“等深”为划分规则对数值维进行划分，这里，Tiny 直方图的划分规则是可选的，这使得我们可以

用其它划分方法替代“等深”划分，使得 ETHs 具有可扩展性。图 1 是 ETHs 的构造过程。

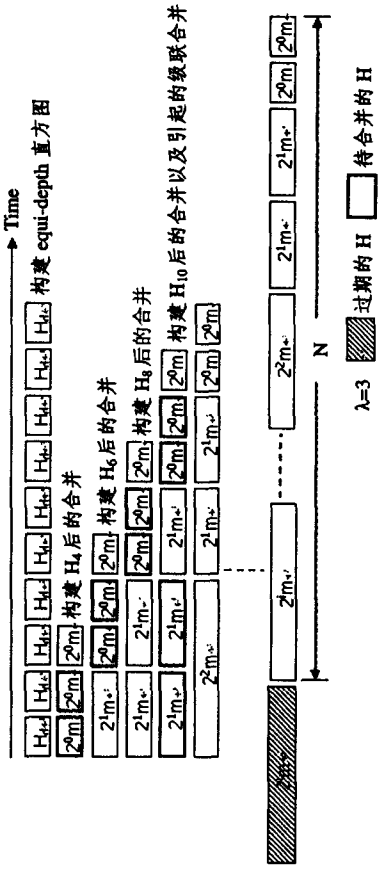

图 1　ETHs 的构建

我们将数据流上到来的元组放入队列 q，假设 q 的长度为 $m=2^k$，($m<<N$)，当队列满时，我们为队列 q 构建一个等深直方图 H_d，令其为 H_0，将队首元组的时间戳作为 H_0 的时间戳，H_0 中的元组数为 $2^0 m$。然后将队列 q 清空，等待下一次队列满时再进行直方图 H_d 的构建。每构建一个新的 H_d，都需要对已存在的直方图进行检查。若元组数为 $2^i m$ 的 H_i 的数目到达阈值 λ(参见 4.2 节)，则将其中最老的两个 H_i 合并，这可能引起元组数为 $2^{i+1}m$ 的 H_{i+1} 的数目超过阈值 λ，从而引起级联合并。若所有子区间包含的元组总数超过 N，或占用的空间总和超过预定值，则将最老的两个子区间合并。

4.5　子区间的合并

假设两个待合并子区间的直方图为 H_1^1 和 H_1^2，它们的元组数均为 $2^i m$，近似度为 ε。本节中采用一种新的合并技术，使得合并后的直方图 H_{i+1} 在 $2^{i+1}m$ 个元组上的近似度为 ε。这是保证我们最终得到的 $ETHs$ 的近似度为 ε 的关键技术。算法 1 描述了这个合并过程。4.3 节中我们将 H 表示为 $\{(b_1,\Delta_1,f_1),(b_2,\Delta_2,f_2),\cdots,(b_\beta,\Delta_\beta,f_\beta)\}$，这里我们同样沿用这个定义。

算法 1　MergeH

输入：$H_1^1:\{(b_i^1,\Delta_i^1,f_i^1):1\leqslant i\leqslant\beta,b_i^1\in V\}$，

$H_1^2:\{(b_i^2,\Delta_i^2,f_i^2):1\leqslant i\leqslant\beta,b_i^2\in V\}$，$l$

// l 为 H_1^1 包含的元组数，H_1^1 和 H_1^2 包含的元组数相同。

输出：$H_{i+1}:\{(b_i^m,\Delta_i^m,f_i^m):1\leqslant i\leqslant\beta,b_i^m\in V\}$

Begin

1：$H_{i+1}:=\Phi$；

2：将 H_1^1 和 H_1^2 合并，并对 b_i 排序，得到$\{(b_i,\Delta_i,f_i):1\leqslant i\leqslant 2\beta,b_i\in V\}$；

3：for i=1 to β do

4：　　$b_i^m=b_{2i},\Delta_i^m=\Delta_{2i},f_i^m=0$；

5：end for

6：for i=1 to β do// 调整桶边界 b_i^m

7：　　$r=(2*l/\beta)*i$；

8：　　计算 b_i^m 在 H_1^1 和 H_1^2 的合并区间中的位置 s；

9：　　if $|s-r|\leqslant 2\varepsilon l$ then

10：　　计算 f_i^m 的值；

11：　　else　　// 移动桶的边界

12：　　找到 b_i^m 左右使得 $|s-r|\leqslant 2\varepsilon l$ 的元素；

13：　　计算相应的 Δ_i^m 和 f_i^m 的值；

14：　　end if

15：end for

16：取 H_1^1 和 H_1^2 中较老的时间戳作 H_{i+1} 的时间戳；

End

4.6　从 ETHs 中获取概要

从图 1 中，我们可以看出，ETHs 是由一组按时间顺序排列的直方图组成的，因此从中计算最近 n 个元组的概要是很容易的，只需要将最近 n 个元组所包含的区间找到，取各个 Tiny 直方图上满足条件的统计数据之和，即可得到最终结果。图 2 给出了一个计算概要的简单例子。

图 2　从 ETHs 中获取概要

4.7　算法分析

在算法 1 中，我们用 $|s-r|\leqslant 2\varepsilon l$ 保证了桶边界的近似度，使其对于 $2l$ 个元组的区间的近似度仍为 ε。在指数划分中，最近 N 个元组，子区间的数目为 $O(\frac{\log(\lambda N)}{\lambda})$，因此，在 ETHs 中，对于最近 $2^k N$ 个元素，子区间的数目为 $O(\frac{\log\lambda N-k}{\lambda})$，这里，$\lambda=\lceil 1/\varepsilon\rceil/2+1$，而每个 $Tiny$ 直方图所占用的空间为 $O(\beta)$，因此 $ETHs$ 的空间复杂度为 $O(\frac{\log\lambda N-k}{\lambda}*\beta)$。

5　实验结果

我们采用平均分布和正态分布数据作为测试数据，构建最近 1 万、10 万、100 万个元组的 ETHs，估算不同选择度的平均误差。并与文[6]中的算法进行比较。我们的测试环境为：CPU 奔四 2G，内存 512M，硬盘 80G，操作系统为 Windows XP。

我们首先测试平均分布的数据，通过改变指数划分的阈值来观察划分粒度对整个估算精度的影响。从图 3 和图 4 可以看出，无论是平均分布还是正态分布，阈值越小，平均估算精度越低。但是，有趣的是，在选择度较低的情况下，如 20%，平均分布和正态分布下的直方图呈现出不同的效果。对于较低的选择度，平均分布下，不同阈值的直方图，其估算精度是不规则的，而在正态分布下，估算精度与阈值的关系是规则的。这实际上是由于 Tiny 直方图的“等深”划分引起的。等深直方图能够更好地反映正态分布的变化，因此，正态分布下，阈值越小，指数划分的粒度越粗，平均估算精度也就越低。而当选择度增大时，两种分布下的直方图都体现出指数划分的特点，即阈值越大，对时间域的划分越优。

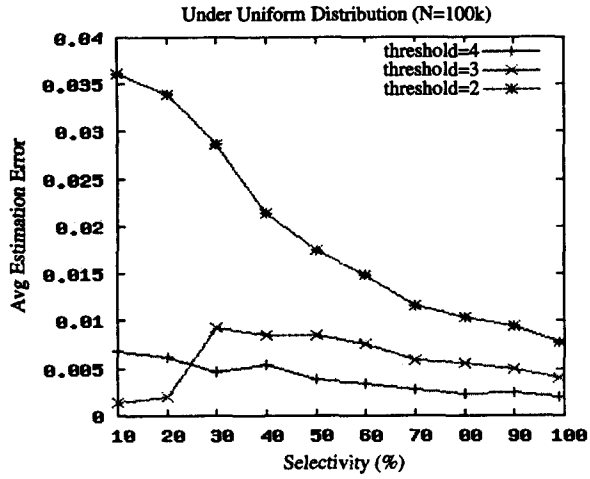

图 3　平均分布下不同指数划分的误差变化

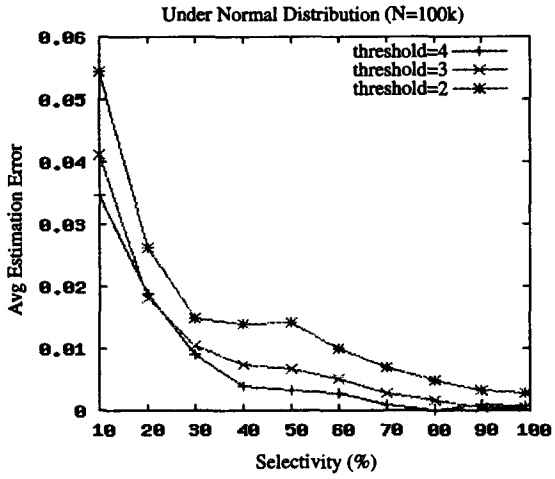

图 4　正态分布下不同指数划分的误差变化

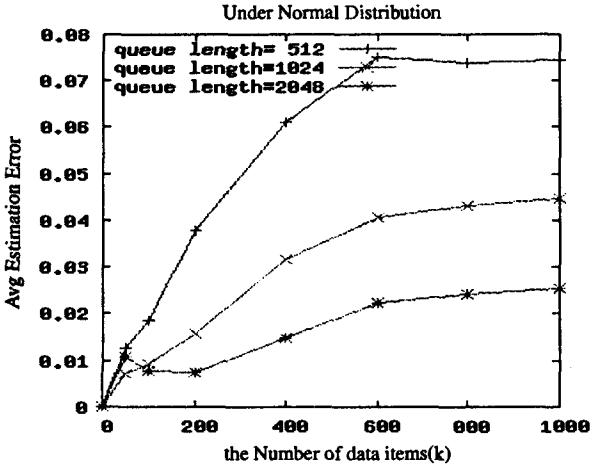

图 5　随数据规模、队列长度变化的误差变化

图 5 是不同数据规模下 ETHs 的平均估算误差。我们基于不同队列长度构建 Tiny 直方图，可以看出，数据规模越大，越适合用较大的队列来构建 ETHs。另外，当数据规模超过 50 万时，平均估算误差趋于平缓，这说明 ETHs 符合我们的设计目标，适合计算数据流上的大纲。

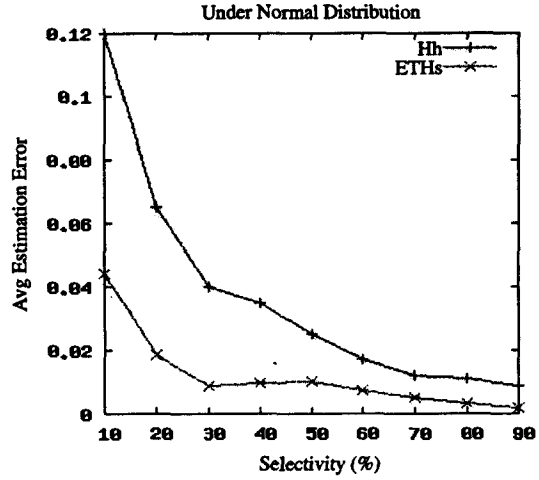

图 6　算法比较

图 6 是 ETHs 与文[6]中的二维直方图 Hh 的比较。在正态分布下，ETHs 的平均估算精度优于 Hh。而且，由于 ETHs 是在指数划分的基础上构建 Tiny 直方图，而 Hh 是对值域划分后再构建指数直方图，因此，ETHs 的空间开销和计算开销都比 Hh 小。

结论和将来的工作　随着数据流应用的日益广泛，与之相关的各种技术也成为研究热点，其中数据流大纲的维护，向我们提出了新的挑战，我们需要用新的思路和方法来解决数据流上滑动窗口这一特殊结构的大纲维护。目前为止，大多数的算法停留在对数据流的增量维护上，本文提出的 ETHs 算法，既能精确地计算增量数据，又考虑到数据流的时间特性，对减量流采用指数划分技术进行衰减处理，同时由于使用等深直方图作为每个子区间的概要结构，从而能够保证误差范围在 ε 之内，在计算时间和存储空间上的代价都很低。下一步，我们需要进一步改进算法在适应数据分布变化上的能力，使得算法能够在实际的数据流上获得更好的性能。

参 考 文 献

1　Manku G S, Rajagopalan S, Lindsay B G. Approximate Medians and other Quantiles in One Pass and with Limited Memory. In: SIGMOD Conf. 1998. 426～435

2　Manku G S, Rajagopalan S, Lindsay B G. Random Sampling Techniques for Space Efficient Online Computation of Order Statistics of Large Datasets. In: Proc. ACM SIGMOD Intl. Conf. on Management of Data, 1999. 251～262

3　Greenwald M S, Khanna S. Space-Efficient Online Computation of Quantile Summaries. In: Proc. of the 2001 ACM SIGMOD Int'l Conf. on Management of Data. Santa Barbara: ACM Press, 2001. 58～66

4　Guha S, Koudas N. Approximating a Data Stream for Querying and Estimation: Algorithms and Performance valuation. In: Proc. of the 16th ICDE Conf. 2002. 567～576

5　Datar M, Gionis A, Indyk P, Motwani R. Maintaining stream statistics over sliding windows. In: Proc. of the 13th Annual ACM-SIAM Symp. on Dis-crete Algorithms, 2002. 635～644

6　Qiao L, Agrawal D, Abbadi A E. Supporting sliding window queries for continuous data streams. In: Proc. 15th Int. Conf. on Scientific and Statistical Database Management, 2003. 85～94

7　Ioannidis Y E, Poosala V. Histogram-based approximation of set-valued query-answers. In: Proc. of the 1999 Intl. Conf. on Very Large Data Bases, 1999. 174～185

8　Poosala V, Ioannidis Y, Haas P, Shekita E. Improved histograms for selectivity estimation of range predicates. In: Proc. of ACM SIGMOD Conf., 1996. 294～305

9　Piatetsky-Shapiro G, Connell C. Accurate Estimation of the Number of Tuples Satisfying a Condition. In: ACM SIGMOD Intl. Conf. on the Management of Data, 1984. 256～275