



一种流数据立方体分析挖掘框架^{*}

金苍宏¹, 刘泽民¹, 吴明晖², 应 晶^{1,2}

(1. 浙江大学计算机科学与技术学院 杭州 310027; 2. 浙江大学城市学院 杭州 310015)

摘 要:流数据是目前一种重要的数据展现形式,对流数据进行 OLAM(联机分析挖掘)操作可为分析人员提供多层次的数据视图。但 OLAM 要求在不同粒度中实现对数据的聚合操作,而流式数据内含时态特性和持续到达特性,使得数据无法被多次重复操作。使用传统 OLAP(联机分析处理)方法无法生成部分物化视图且流数据规模宏大,受限于存储空间大小而无法保存全部数据单元信息。针对上述问题,提出了一种基于概要技术的流数据 OLAM 框架——sketch cube(概要立方体),该框架把任意维度组合映射成唯一自然数,根据上下限单调原则对维度组合裁剪,在类线性空间中保存有效数据单元信息,并构建时间序列索引提高检索效率。通过理论分析给出使用 sketch cube 的前提条件,同时通过真实海量流数据实验分析表明,sketch cube 在有效性、存储空间效率和正确率上可以满足实时挖掘的需求。

关键词:流数据;概要立方体;联机分析挖掘;实时分析

doi: 10.3969/j.issn.1000-0801.2014.09.009

A Cube Analytical Mining Framework for Stream Data

Jin Canghong¹, Liu Zemin¹, Wu Minghui², Ying Jing^{1,2}

(1. College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China;

2. Zhejiang University City College, Hangzhou 310015, China)

Abstract: Stream data has been one of the most significant data format recently. OLAM (online analytical mining) operation could provide multi-level data views for analysts. However, OLAM operations depend on data aggregation, which is in conflict with the continuous incensement and dynamic nature of stream data. Thus, partial materialized view from stream data directly by typical OLAP approaches cannot be created and all data cells for the limitation of storage cannot be saved. In order to solve the above problems, an advanced sketch based OLAM framework named sketch cube to analyze stream data was proposed. Sketch cube maps a set of attributes to a unique number and stores it in sub-linear data structure, and then builds an inverted index by tiled time window. The precondition of using sketch cube by theoretical analysis was given and the storage efficiency and query performance on mass mobile data corpus was evaluated, which supports requirements of real-time analysis.

Key words: stream data, sketch cube, online analytical mining, real-time analysis

1 引言

联机分析挖掘(online analytical mining, OLAM)是一种

结合数据挖掘和联机分析处理(online analytical processing, OLAP)的知识发现方法。使用 OLAM 可在多维数据库的不同数据子集和抽象层次上进行探索式挖掘,因

^{*} 浙江省自然科学基金资助项目(No.LQ14F020002),浙江省本科院校中青年学科带头人学术攀登基金资助项目(No.PD2013453)



此在决策定制、市场分析、可视化等领域都发挥着重要作用。随着移动互联网的迅速发展和广泛应用,各种类型的数据流规模急剧增大,如金融流数据、图像流数据、社交流数据等。挖掘数据流中包含用户刻画和行为动作的相关信息,成为当前研究的一个热点^[1]。如在移动应用分析系统中,通过 SDK(软件开发工具包)可以采集应用的名称、使用时间、GPS 信息、IP 地址、手机型号、网络类型等多个维度的信息。对此类信息进行 OLAM 分析,可提供灵活的多角度数据分析场景,提高商业竞争能力。例 1 给出一个流数据 OLAM 的分析场景。

例 1:运营方希望能实时监控某款应用在不同人群中的活跃度,以便其能分析出核心用户,实时调整留存和激活策略,增强其竞争力。

对于不同人群的划分可以使用数据立方体分析,可以在多个子空间中统计相关活跃度。但是,流数据处理具有单次扫描、较低的时间和空间复杂度、适应动态变化的流速等要求,且系统挖掘时效性强,需实时返回分析结果,而事先又无法指定所需要观察的维度组合,需要对维度组合进行全量分析,因此实时计算量巨大。

基于上述的要求,使用现有的 OLAM 方法,把数据仓库中的记录通过 OLAP 投射到不同空间生成数据单元,再通过多角度综合分析数据的方法^[2]无法满足上述要求,原因如下。

(1)数据量大

和传统的数据仓库相比,移动互联网下的流式数据是海量的且增长迅速。传统 OLAP 工具在存储空间和计算效率上都无法满足实时性能要求。

(2)数据动态性强

传统 OLAP 方法,通过对数据立方体建立物化视图(materialized view)可以预计算部分结果,从而提高实时查询的效率,达到空间和效率上的平衡^[3],如 QC-Tree、侏儒立方体等方法^[4]。但流式数据是动态可变的,因此对于数据无法做静态的预处理,必须要实时计算。

(3)数据时效性强

传统 OLAP 框架处理的是归档数据,对数据的实效性要求不高。而数据流内含的时态特征表明,数据是有生命周期的。用户往往只对特定时间窗口内的数据感兴趣,而忽略窗口外的数据。如例 1 中,只需要统计分析促销活动过程中的最近的数据,而忽略其他历史归档数据。

由此可见,传统的 OLAM 方法不能很好地处理数据

流。而其他基于数据抽样^[5,6]和基于数据压缩^[4]两种流式数据处理方法受限于计算能力和存储空间的大小,只在整体空间或几个分组中进行数据分析,如 stream^[7]、TelegraphCQ^[8]、SMM^[9]等。但它们没有提供在任意子空间或任意层次中对数据进行分析的方案。

本文提出一种流数据立方体处理框架——sketch cube(概要立方体)。并在 storm 开源流处理平台中实现。sketch cube 在较小的空间中实时聚合有效数据单元信息,可满足 OLAM 的分析需求,其贡献在于以下几点。

- 提出一种可扩展的数据单元标识方法,对流数据的任意维度组合通过配对函数(pairing function)^[10]映射生成唯一标识且保证维度扩展后数据仍不冲突。
- 提出一种改进概要统计模型——ACM(advanced count model),ACM 根据数据分布特性和存储模型的设计特征,可有效裁剪长尾的数据单元,以提高计算能力和存储空间效率,同时可改进数据单元统计精确度。
- 提出一种基于时间窗口的倒排检索模型,可支持任意时间粒度的组合。在类线性存储空间中存放所有有效数据单元的统计信息,最大限度地保证了信息的完整性。
- 从理论上给出 sketch cube 的适用场景和准确率范围。通过 storm 实现 sketch cube 功能,且在海量真实数据流下测试模型在不同参数和时间片长度下的吞吐量和准确率。理论分析和实验证明 sketch cube 的查询效率高且能支持流式数据下的立方体全量分析。

本文研究主要结合传统的 OLAP 方法和流数据处理方法。

已有的对传统 OLAP 的扩展研究如(参考文献[11~13])引入了文本属性,提出文本立方体(text cube)概念。Lin C 等^[13]提出基于关键词的数据立方体查询,对数值和文本维度分别建立层次结构,通过倒排索引检索出相应的数据单元。在此基础上,Ding B 等^[12]提出基于关键词的 top- k 数据单元查询问题,通过对关键词的信息检索(information retrieval)度量,提取出与关键词最相关的 k 个数据单元。与此相反,参考文献[11]提出了数据单元主题判定问题,对给定层次的数据单元结合自然语言模型,判断其所属的主题和相关概率。

流数据结合 OLAP 操作产生了流立方体(stream cube)

的概念。参考文献[4]使用倾斜时间框架(tilted time framework)管理数据,使用线性回归函数对流数据在不同维度上进行压缩。但是该方法物化数据带有主观性,会丢失部分信息。同时在上下层节点合并时需要频繁更新 head 表,更新速度较慢。文本流数据立方体(如 Liu X 等^[14])通过 Twitter stream 数据,建立热点图和情感图等应用。该方法虽然使用时间序列分析,但还是把信息看成静态地存放在数据仓库中而进行分析。其他的流数据分析方法,如抽样方法^[6,15],通过建立不同的分布概率函数(probability distribution function, PDF)对异构源数据进行抽样和语义分析,获得不同的权重,通过样本值的上下限来估计实际的聚合值。方法依赖于样本的采集和分布情况,因此对于错误率无法进行有效的控制。参考文献[16]的方法则使用压缩策略,通过对数据特征分析,只保留其主要元素,而过滤其他元素。但它们把数据看成整体,挖掘不同数据流和不同时间段之间的关联关系,而不是有层次的可以进行 OLAP 操作的模型。

此外,计数型散列技术统计数据的方法有 naïve counting bloom filter (NCBF)、d-left counting bloom filter (dlCBF) 和 binary shrinking d-left counting bloom filter (BSdlCBF)等^[17]。在此基础上,概要技术^[18,19]可在二阶矩阵大小空间内保存流数据,把 N 个输入元素压缩在 $\ln N$ 的存储空间中。

2 问题描述

2.1 基本概念

定义 1 (多维数据流模型 (multi-attribute stream data model, MSDM)) 定义某时间点上的数据流记录。MSDM 为一个二元组结构 (S, T) , 其中, S 为多维度数据模型 $S=(A_0, A_1, \dots, A_i, M)$, A_i 表示标准属性, M 为度量维度, T 为某个时间点。

如 $((\text{Wi-Fi}, \text{hz}, \text{Samsung}; 1), 20140510081559)$ 中, $S=(\text{Wi-Fi}, \text{hz}, \text{Samsung}; 1)$ 为维度模型, hz 表示“杭州”, 20140510081559 为时间点, 1 为度量值。

定义 2 (流立方体) 表示给定一个时间段内的 MSDM, 按不同的维度构建的一个数据立方体 stream cube $= (A_1, A_2, \dots, A_n, M, T)$ 。对于 stream cube 中的每条数据 $r=(a_1, a_2, \dots, a_n, m, t)$, 其中 $r[A_i]=a_i \in A_i$, a_i 为属性 A_i 的值, m 为度量值。对于时间维度 T 有 $t_{i-1} < t_i$ 表明数据是按时间前后连续到达。

如例 1 中, 给定时间间隔为 5 min, 在该时间段内收集

的所有数据产生的数据立方体, 即该时间段内的流立方体。

定义 3 (祖先单元和后代单元) 对于流立方体中的数据单元 C_m 和 C_n , 定义 $*$ 表示该维度折叠且计算不考虑, 则 C_m 是 C_n 的祖先 (C_n 是 C_m 的后代) 可表示为:

$$\begin{cases} \exists i: C_m[A_i] = *, C_n[A_i] \neq * \\ \forall j: C_m[A_j] = C_n[A_j], i \neq j \\ C_m[t] = C_n[t] \end{cases} \quad (1)$$

记为 $C_n[t] \subset C_m[t]$, 即两个数据单元之间的时间相同, 祖先单元至少在一个维度上包含子孙单元且在其他所有维度上和子孙单元的值相等。

对于给定的流数据单元 $D_0=((\text{Wi-Fi}, \text{hz}, \text{Samsung}), 20140510081559)$ 的父节点有如下 7 个:

- $D_1=((*, \text{hz}, \text{Samsung}), 20140510081559)$;
- $D_2=((\text{Wi-Fi}, *, \text{Samsung}), 20140510081559)$;
- $D_3=((\text{Wi-Fi}, \text{hz}, *), 20140510081559)$;
- $D_4=((*, *, \text{Samsung}), 20140510081559)$;
- $D_5=((*, \text{hz}, *), 20140510081559)$;
- $D_6=((\text{Wi-Fi}, *, *), 20140510081559)$;
- $D_7=((*, *, *), 20140510081559)$

上述例子中没有显示数据单元的度量值 M , 祖先单元的度量值可以由其子孙单元的度量值通过聚合操作而计算得出。聚合操作可以分为分布型(distributive)、代数型(algebraic)和整体型(holistic)3 类^[20]。

2.2 流数据立方体问题描述

在一组给定的持续的 MSDM 中, 按时间窗口大小, 划分出不同的数据片段, 按照其维度生成流数据立方体。流数据立方体实时提供了清洁、有组织和汇总的数据, 因此可以在不同粒度父子数据单元中分析挖掘。通过和流数据框架的结合, 大大增强探索式数据挖掘能力和灵活性。如例 1 所示, 可以满足实时的、全量的多层次数据挖掘的需求。

3 sketch cube 框架结构

本节给出构造 sketch cube 框架所需的主要部件: 数据单元标识算法及其改进模型、流数据立方体存储结构和裁剪模型、时间窗口索引等。

3.1 数据单元标识算法

流数据立方体产生了海量的维度组合, 即数据单元, 这些数据单元中的不同属性值包含多种类型, 有连续型、



离散数值型、字符型等。需要对这些属性值进行字典化处理,并给每个数据单元一个唯一的标识,用于后续计算。

数据单元标识(data cell identifier, DCI)算法的功能就是把不同的维度组合映射成唯一整数,且算法支持维度的扩展,即新增维度或修改维度值都不会与原有映射值冲突。在流数据 $R=(A_1, A_2, \dots, A_n, M)$ 中, $|A_i|$ 表示第 i 维的基数。使用两个步骤进行数据单元标识。

(1) 由于维度值的多样性,需要先把记录 R 的维度 A_i 映射成连续的自然数,即:

$$F_{\text{map}}(a_i) \rightarrow N_i, 0 \leq N_i \leq |A_i| \quad (2)$$

(2) 在步骤(1)中产生 n 个自然数 N_i , 形成集合 S , 使用配对函数对 S 中的任意非空子集生成唯一的自然数。

配对函数的定义为把两维度元组映射成一维度元组的函数 $N \times N \rightarrow N$ 。它是一个双射函数,其单调递增的特性如式(3)所示,保证其不会重复。

$$p(x, y) < p(x+1, y); p(x, y) < p(x, y+1) \quad (3)$$

对于高于两维度的元组可以使用嵌套模式进行,文本使用康托尔配对函数(Cantor tuple function),如图1和式(4)所示,按维度顺序进行嵌套配对,把中间结果当成下一步操作的输入。

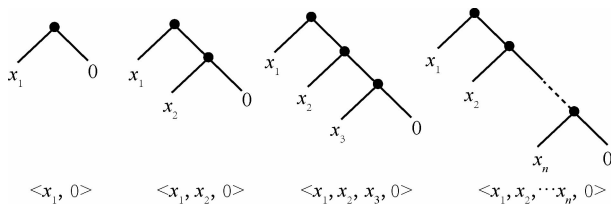


图1 有限元素算术图

$$\begin{cases} \pi^1(x) := x; \\ \pi^{k+1}(x_1, \dots, x_{k+1}) := \pi(\pi^k(x_1, \dots, x_k), x_{k+1}) \end{cases} \quad (4)$$

3.1.1 改进配对函数

康托尔配对函数用嵌套方式生成映射值,当元组维度较高或者某些维度基数较大时,会导致巨大的映射值。提出改进配对函数(advanced pairing function, APF)模型,其思想是在不改变计算模型的同时,产生较小的映射值。其理论依据如下:

- 在数据立方体中,维度的顺序与数据单元描述无关,即数据单元 $(A_1, A_2) = (A_2, A_1)$;
- 在嵌套方法中,越早进入计算的数值参与循环的次数越多,对结果值大小的影响越大。

APF模型定义如式(5)和式(6)所示。

$$APF(A_1, A_2, \dots, A_n) \rightarrow |A_1| > |A_2| > \dots > |A_n| \quad (5)$$

对给定属性 A_i :

$$\text{freq}(a_{i1}) > \text{freq}(a_{i2}) > \dots > \text{freq}(a_{in}) \rightarrow n_1 < n_2 < \dots < n_n \quad (6)$$

如图1所示,嵌套计算从尾部开始,式(5)表示输入配对函数的维度顺序由维度基数大小决定,基数大的维度靠前。式(6)表示把维度值映射成连续自然数时,对出现频率高(freq 值大)的维度值赋给小自然数。维度基数的大小和维度值的频率可以由统计或经验获得。如本文实验中,手机操作系统维度基数小于城市维度基数,Android机型多于苹果机型。使用APF模型可以有效地减少大数值在递归中的计算次数,达到最终控制输出映射值的目的。

3.1.2 算法步骤

结合APF模型,给出算法1数据单元标识(DCI)算法的步骤。DCI算法先把流数据中每条记录的维度值映射成由小到大的自然数(行①),通过递归函数得到与之相关的所有维度组合(行②),并使用康托尔配对函数获得所有组合唯一标识(行③~⑦)。

算法1 数据单元标识算法

输入:流数据记录 $r=(a_1, a_2, \dots, a_i, m, t)$

输出: r 所有维度组合标识

- ① 根据APF模型中式(5)和式(6),将记录 r 中的维度映射成 $(n_1, n_2, \dots, n_n)$
- ② $\text{set} \leftarrow \text{combination} = (n_1, n_2, \dots, n_i)$ 的所有组合, $\text{size of combination} = 2^i$ 。
- ③ $\text{set} \leftarrow \text{result} \leftarrow \{\}$;
- ④ for all $x \in \{\text{combination}\}$ do
- ⑤ $\text{set} \leftarrow \text{result} \leftarrow \text{pairing function}(x)$;
- ⑥ end for
- ⑦ return $\text{set} \leftarrow \text{result}$;

3.2 存储结构描述

提出一种通过 sketch 技术对流数据单元进行统计的方法,首先给出模型所需的独立散列族定义。

3.2.1 独立散列族定义

定义4 (均匀散列函数) 对于给定的元素集合 U 通过散列函数 $h(x)$ 投射到 n 个元素中,对每个 $j=1, \dots, n$, 给定 $x \in U$, 有 $P[h(x)=j]=1/n$, $h(x)$ 则为均匀散列函数。

定义5 (互相独立散列函数族) 在 F 散列族中任意选择 a, b 两个随机函数,如果 $P[h_{ab}(x)=i]=\frac{1}{n}, \forall x \in U, i \in [n]$ 且 $P[h_{ab}(x)=h_{ab}(y)] \leq \frac{1}{n}, \forall x, y \in U$, 则 F 为相互独立的散列函数族。

3.2.2 计数最小概要模型

计数最小概要(count-min sketch, CM sketch)模型^[10]是一个使用相互独立散列函数族函数统计流数据元素出现频率的模型。其结构如图 2 所示,是一个 $d \times w$ 的二维数组。

$$\begin{bmatrix} \text{sketch}[1,1] & \cdots & \text{sketch}[1,w] \\ \vdots & \ddots & \vdots \\ \text{sketch}[d,1] & \cdots & \text{sketch}[d,w] \end{bmatrix}$$

图 2 CM sketch 存储结构

其中, d 表示相互独立散列族函数的个数, w 表示每个散列函数的映射范围,如式(7)所示:

$$h_k: \{1 \cdots N\} \rightarrow \{1 \cdots w\}, 1 \leq k \leq d \quad (7)$$

显然, n 个元素可以表示 2^n 个两两组合元素集合。因此,设计包含 d 个函数的相互独立散列函数族,只需用 $\sqrt{d}$ 个不同元素,如式(8)所示:

$$\text{SeedSet} = \{1, 2, \dots, n \ln \sqrt{d} + 1\} \quad (8)$$

从 SeedSet 中随机取不同的元素 a 和 b , 定义散列函数方法如式(9)所示:

$$h_{a,b}(z) = a \times z + b \quad (9)$$

由 $h_{a,b}(z)$ 产生的散列值是变长的,式(10)把值规约到 w 大小的数组中。

$$h_{a,b}(z) = ((a \times z + b) \% p) \% w, p \text{ 为大质数} \quad (10)$$

3.2.3 CM sketch 效率

对于第 t 次到达的元素 c , 计数最小概要模型更新操作如式(11)所示:

$$\forall 1 \leq j \leq d: \text{CM}[j, h_j(i)] \leftarrow \text{CM}[j, h_j(i)] + c_i \quad (11)$$

即使用不同散列函数找到存储结构中的下标位置,并添加相应值,其更新时间复杂度为 $O(\log \frac{1}{\delta})$ 。

统计元素 a_i 在 CM sketch 中的估计值 $\hat{a}_i$ 的操作如式(12)所示:

$$\hat{a}_i = \min_{1 \leq j \leq d} \text{CM}[j, h_j(i)] \quad (12)$$

即通过散列函数族中的每个函数计算该元素在对应数组中的下标值,获得所有可能值中的最小值即该元素的估计值,其查询时间复杂度为 $O(1)$ 。

流数据有收款机模型(cash register model)和十字转门模型(turnstile model)两种模型。计数最小概要模型只支持收款机模型,即统计值只能递增,因此基于此的 sketch cube 只支持分布型聚合操作。对于 $(A_1, A_2, \dots, A_n)$ 数据立方体,所有数据单元个数 $T = \prod_{i=1}^n (|A_i| + 1)$ 。则 sketch cube 模型的

压缩率 P 为式(13)所示:

$$P = \frac{\prod_{i=1}^n (|A_i| + 1)}{w \times d} \times 100\% \quad (13)$$

3.2.4 模型精度和前提条件

计数最小概要模型中 w 和 d 的值由输入参数决定,分别为 $w = \frac{e}{\epsilon}$ 和 $d = \ln(\frac{1}{\delta})$, 假设输入元素的集合基数为 $\|a\|$, 则对于元素 i 的估计值满足不等式(14)的概率至少有 $1 - \delta$ ^[22], 其中 ϵ 参数控制估计值大小。

$$\hat{a}_i \leq a_i + \epsilon \|a\| \quad (14)$$

式(14)给出了预测元素 a_i 统计值的偏差范围,即预测统计值 $\hat{a}_i$ 小于实际值 $a_i + \epsilon \|a\|$ 的概率为 $1 - \delta$, 可以根据元素多样性和期望精确度调整 ϵ 和 δ , 进而调整存储模型中的 w 和 d 值。

计数最小概要模型是一个数据敏感模型,对于偏斜数据(skew data)的支持较好(Zipfian 参数 $1 < z < 3$)。而对于均匀分布的数据,在 w 不是足够宽的条件下,易产生较大的误差。因此,对于这类分布的数据,计数最小概要模型并不适用。

3.2.5 计数平均最小概要模型

对于分布较为均匀的数据集,不同元素之间的冲突影响较大,可使用计数平均最小概要(count mean min sketch, CMM sketch)统计个数。CMM sketch 假设元素均匀分布在散列数组中,则对于元素 c_i 的噪音因子定义如式(15)所示:

$$\text{noise}(a_i) = \frac{\|a_i\| - \text{sketchcount}(a_i)}{w - 1} \quad (15)$$

元素的预测值需要减去噪音因子,元素概要预测值返回结果取各散列数组中的值的中位数,如式(16)所示:

$$\hat{a}_i = \text{Md}(\text{sketchcount}(a_i) - \text{noise}(a_i)) \quad (16)$$

使用上述哪种模型进行统计,由流数据分布特征和业务需求而定。在本文实验中,由于数据单元标识产生的维度组合值是偏斜的,因此采用计数最小概要模型。

3.3 裁剪模型

由数据单元标识模型产生的维度组合是幂级增长的,如给定 n 维记录 $r = (a_1, a_2, \dots, a_n, m, t)$ 可以产生 2^n 种维度的组合。随着维度数量增加,产生的数据单元数量是巨大的。海量的维度组合会导致计数最小概要模型的冲突率变大,影响精确度。因此,需要对维度组合进行裁剪。

在数据立方体不同聚合操作类型中,对分布类型的操



作如 sum 和 count , 祖先单元 (低维数据单元) 聚合了其下的所有后代单元 (高维数据单元) 的值。由此可得后代数据单元的上限 (up bound) 和 下限 (low bound), 从而可以推理出祖先单元值的范围 (up bound-low bound), 如式 (17) 所示:

$$\text{count}(X) = \text{sum}(\{\text{count}(X_i | i=1, \dots, n)\}) \quad (17)$$

其中, 祖先单元 X 可以由最小单元 X_i 集合组成 MSP (most specific partition)^[13]。对于数据单元 X 下的任意子空间 g 的上下限为式 (18) 所示:

$$\begin{aligned} \overline{\text{count}(g)} &= \text{sum}(\{\text{count}(X_i | i=1, \dots, n)\}) \\ \underline{\text{count}(g)} &= \min(\{\text{count}(X_i | i=1, \dots, n)\}) \end{aligned} \quad (18)$$

因计数最小概要模型适合于数据浓度大的元素 (如大于阈值 τ)。如图 3 所示, 结合式 (18), 数据单元间的包含关系在 count 操作中体现出单调性, 即 $C_a[\text{count}] > \tau \rightarrow C_a[C_a[\text{count}]] > \tau$, $C_c[\text{count}] > \tau$ 。其他较复杂代数聚合操作如 avg 的上下限不等式在参考文献[13]中有所定义。

裁剪规则: 对于分布型聚合操作, 如祖先单元的测量值小于阈值, 则所有它的后代单元的测量值必定小于阈值, 因此对于计数最小概要模型可以裁剪掉所有后代单元。

3.4 改进统计模型

由于 sketch cube 使用计数最小概要模型, 针对分布型聚合操作, 结合裁剪规则, 提出改进统计模型 (ACM)。模型通过预测单维数据单元的测量值, 达到减少维度组合的目的。如果测量值小于阈值, 可以删除所有包含该维度值的子孙单元。如在记录 $r=(a,b,c)$ 中, 若 $C_a[\text{count}] < \tau$, 则 $C_{abc}[\text{count}] < \tau$ 。对于 n 维度的流数据中, 先进行 n 次单维度统计, 假设有 k 个单维数据单元测量值小于阈值, 则可以裁剪的维度为 2^k , 在对于每条流数据中可减少统计操作 $2^k - n$ 次。因为流数据中的维度值往往是稀疏的, 因此裁剪模型可以排除大多数的维度组合。如果一维裁剪模型的裁剪效果不佳, 可以使用二维或高维裁剪模型, 方法类似。算法 2 结合裁剪方法的维度生成规则。维度裁剪规则如图 3 所示。

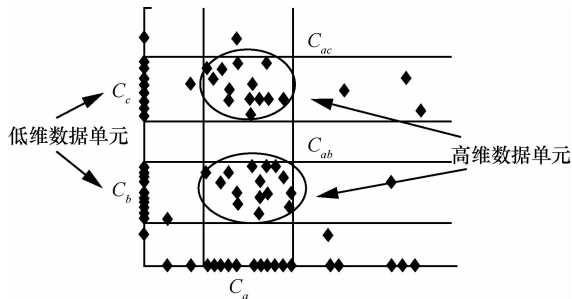


图3 维度裁剪规则

算法 2 改进统计模型

输入: 流数据记录 $r=(a_1, a_2, \dots, a_i, m, t), \tau$

输出: r 所有维度组合标识。

- ① $\text{set} \leftarrow \{1\text{-DCounter}\} \leftarrow \{\};$
- ② $\langle 1\text{-DCounter} \rangle \leftarrow C_{a_k}[\text{Count}] > \tau, 1 \leq k \leq i;$
- ③ $(a_1, a_2, \dots, a_i) \leftarrow (a_1, a_2, \dots, a_i) \setminus \langle 1\text{-DCounter} \rangle;$
- ④ $(a_1, a_2, \dots, a_i) \rightarrow (n_1, n_2, \dots, n_i)$
- ⑤ $\text{set} \leftarrow \langle \text{Combination} \rangle = (n_1, n_2, \dots, n_i)$ 的所有组合;
- ⑥ $\text{set} \leftarrow \langle \text{result} \rangle \leftarrow \{\};$
- ⑦ for all $x \in \{\langle \text{Combination} \rangle\}$ do
- ⑧ $\langle \text{result} \rangle \leftarrow \text{Pairing Function}(x);$
- ⑨ end for
- ⑩ return $\langle \text{result} \rangle;$

本文使用 $\text{ACM}(m)$ 表示模型取每个维度上频率最高的 m 个维度值进行组合, 可以减少计算模型的复杂度和存储的空间, 且能保证最后的高频数据单元不会被裁剪掉。实验给出了使用 ACM 模型在不同 m 参数值下的优化效率。

3.5 时间序列索引描述

流数据具体有内在时序性, 对流数据挖掘用倾斜时间窗口 (tilted-time window, TTW)^[23] 在不同时间粒度 (multiple time granularities) 上进行分析。如图 4 所示, sketch cube 按时间片段对元素组合进行统计把结果放入计数最小概要模型。sketch cube 设计的存储结构可支持任意时间粒度的组合, 其合并计算式如式 (19) 所示:

$$\forall 1 \leq j \leq d: CM[j, h_j(i_{t_1+t_2})] \leftarrow CM[j, h_j(i_{t_1})] + CM[j, h_j(i_{t_2})] \quad (19)$$

对于给定散列函数族, 相同维度组合在不同时间中的映射地址相等, 可单次扫描小颗粒时间累加得到大颗粒时间度量值。

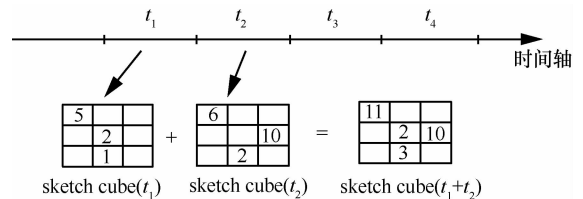


图4 时间窗口聚合操作

4 实验结果与分析

目前已有流处理框架都是先存储数据片段再进行实时的 OLAP 操作, 和文本所提实时计算和预存储模型的 sketch cube 不同。因此本文主要通过如下 3 个方面验证方法的有效性和正确性。

- 通过真实移动应用日志记录分析场景进行实验。分析 sketch cube 框架在不同时间窗口、不同裁剪模型参数和不同存储模型大小下的正确率、剪枝能力、框架吞吐效率;验证参数之间的影响和整体框架的有效性。
- 理论分析和比较 sketch cube 在存储空间上和其他存储模型的差异。
- 描述使用 storm 开源流框架搭建 sketch cube 的实现方式,并给出一个应用场景。

4.1 实验环境和数据描述

实验机器选择 4 台 Dell 服务器,16 核 IntelXeon 227 GHz,内存为 32 GB,硬盘为 4 TB,操作系统为 Cenos 6.3,Java 版本为 JDK7。因为系统每秒接收流数据在 1 万条左右,采用 Kafka 和 storm 框架对数据进行处理,存储模型保存在 MongoDB 中。

数据集合:测试数据集来源于真实的移动嵌入式 SDK 中采集的应用记录,每条记录包含 56 个维度数值。本实验选择其中 8 个有代表性的维度进行 OLAP 统计,使用 5% 的均匀抽样(uniform sampling)方法运行 1 h,其含义和维度值的基数见表 1。

表 1 移动数据信息结构

属性描述	维度基数
终端联网类型字典编号	9
当前应用的版本号	23
屏幕分辨率字典编号	123
操作系统字典编号	133
当前应用的发布渠道编号	165
运营商字典编号	402
城市字典编号	413
移动设备型号字典编号	4 436

对上述 8 个维度进行全立方体组合可得超过 10^{17} 个数据单元。但通过分析可得,数据单元的分布是稀疏且偏斜的。在不同大小的时间窗口中,对维度组合的分布进行 Zipfian 的计算结果见表 2。可得维度组合结果是偏斜的,

因此,本实验数据符合使用 sketch cube 框架的条件。

表 2 抽样移动数据分布

时间窗口	样本大小	维度组合数量	偏斜因子
20 min	321 315	2 795	1.92
40 min	636 795	3 257	1.99
60 min	963 675	3 529	2.05

表 2 还表明数据是线性增加的,维度组合数量(DCI size)随时间增加而趋于稳定,而偏斜因子和时间窗口长短无关。

表 3 给出了实验涉及参数的取值范围和缺省值。对于 sketch cube 所涉及的散列函数使用 MurmurHash V3 产生 d 个相互独立的散列族,时间窗口以 20 min 为基本单位,ACM 选择 m 大小的裁剪模型,且在多组产生结果中进行分析和比较。

表 3 实验参数

实验参数	取值
散列类型	MurmurHash V3
时间窗口/min	20,40,60
CM-sketch 行数(d)	5
CM-sketch 列数(w)	521,1 021,2 027
ACM 参数 m	10, 20, 30, 40, 50
比较结果	20, 40, ..., 200
维度个数	8

表 4 给出了 ACM 模型在 3 个不同参数下和全量计算模型(full count model, FCM)在包括吞吐量、裁剪能力和模型准确率方面的比较结果。

4.2 sketch cube 计算性能

本节讨论 ACM (m) 裁剪方法在不同参数 m 下对 sketch cube 模型的剪枝能力。FCM 表示统计全部维度组合,取 $m=(10,20,30,40,50)$ 5 个值,在 3 个不同时间长度中观察 ACM 的输出变化。如图 5 所示,ACM 模型在不同参数下都具有 10 倍左右的裁剪能力且很显然参数值越小,保留的数据单元越少,裁剪能力越强。

表 4 部分实验结果

模型	sketch 性能数据 集大小=170 880	DCI 大小			MPE (20 min, $w=521$)			MPE (20 min, $w=2 027$)		
		20 min	40 min	60 min	top-20	top-80	top-160	top-20	top-80	top-160
ACM(10)	37 ms	15 896	18 796	20 498	0.081 6	0.142 7	0.259 7	0.006 337	0.014 63	0.021 59
ACM(30)	239 ms	90 592	123 650	145 258	0.168 6	0.359 5	0.636 1	0.024 38	0.056 01	0.104 5
ACM(50)	424 ms	113 719	157 519	187 134	0.188 5	0.412 9	0.723 2	0.029 09	0.067 73	0.124 5
FCM	2 024 ms	654 202	1 026 186	1 551 168	0.552 1	1.307 3	2.084 2	0.123 4	0.266 1	0.436 7

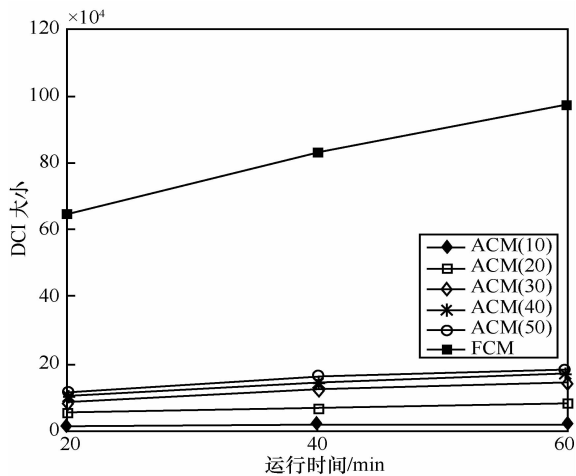


图5 ACM 剪枝实验

接着,需要测试剪枝模型的有效性和可用性,即裁剪后的模型是否仍然可以满足需求,而不会丢失很多有效数据。实验取 20 min 数据,分别统计 ACM 不同参数下提取 DCI 数量和裁剪后维度组合数量。如图 6 所示,ACM 模型参数从 10 增加到 50 过程中,虽然 DCI 的数量有所提高,即计算得到的数据单元数量有明显的增加,但是裁剪后维度组合出现次数却基本相等。这说明数据是偏斜的,大量的数据单元反复出现,而长尾的数据只有少量出现。因此,ACM(10)基本包含了所有高频率的组合,在后续的框架构建和查询中,笔者使用 ACM(10)作为裁剪模型。

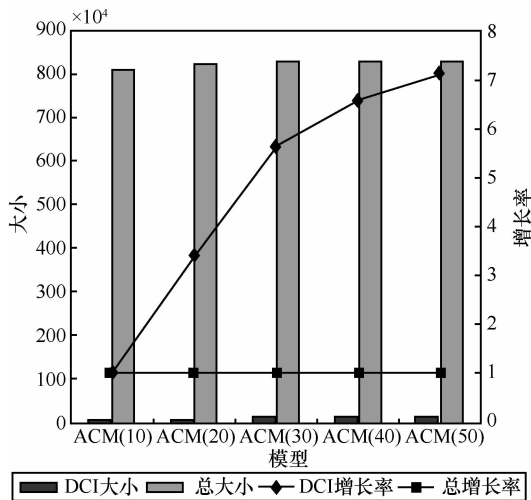


图6 ACM 裁剪模型效率

4.3 存储空间和误差

使用计数最小概要模型的 sketch cube 中保存的元素统计数值要大于元素实际值,本实验使用平均百分比误差 (mean percentage error, MPE) 统计计数的准确率。计算式如下:

$$MPE = \frac{1}{n} \sum_{i=1}^n \frac{f_i - a_i}{a_i} \quad (20)$$

其中, a_i 是实际值, f_i 是预测值, n 是测试数据数量。

图 7 展示 sketch cube 中的存储空间大小和 ACM 参数对 MPE 的影响。取 10 min 数据,在不同存储宽度中比较 4 类模型的 MPE 值。ACM(m) 中 m 越小,裁剪能力越强,模型准确率越高。随着存储空间的增大(即 w 参数的变大),所有模型的准确率都有所提高,这是因为存储空间可以增加减少散列的碰撞概率,从而提高模型的准确率,但 ACM 整体准确率在任意存储模型下都较 FCM 有明显提高。

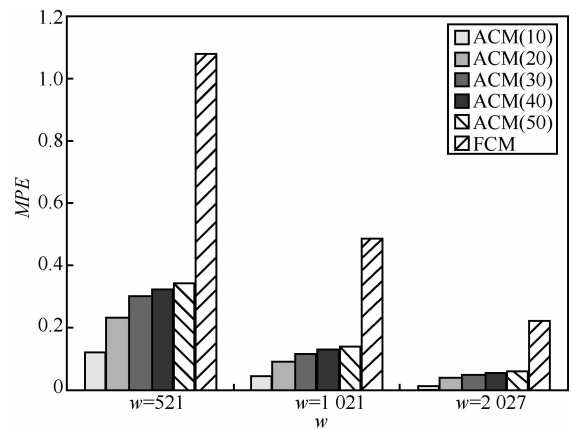
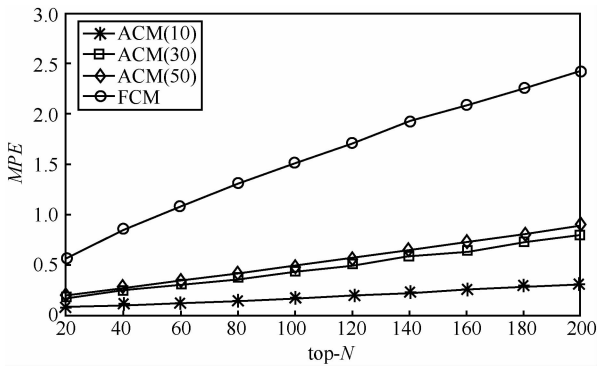
图7 w 和 ACM(n) 对 MPE 的影响

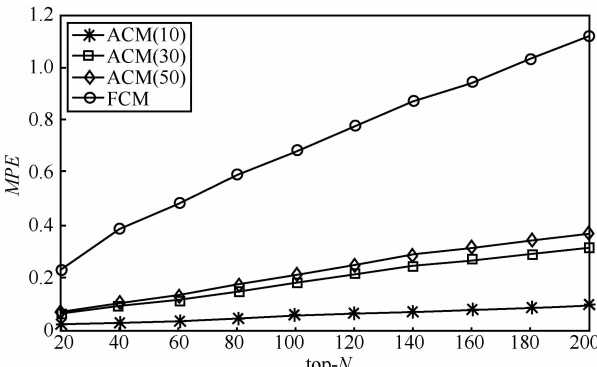
图 8 显示存储空间、结果集大小和 MPE 之间的关系。top- N 从大到小排序,从横轴看,频率越高的数据单元,其 MPE 值越小,准确率越高。如在 1 021 宽度的存储模型中,ACM(10) 模型对前 20 的高频数据单元的统计错误率为 1% 左右,而 FCM 的错误率超过 20%。随着 top- N 的变大, MPE 值也上升,说明 sketch cube 产生的数据单元是偏斜的。在不同的系统应用中,可以选择合适的存储空间大小和计算结果大小,如需要计算的 top- N 较大,则需要使用较小的 ACM 和较大的存储空间 w 。

4.4 吞吐能力和计算时间比较

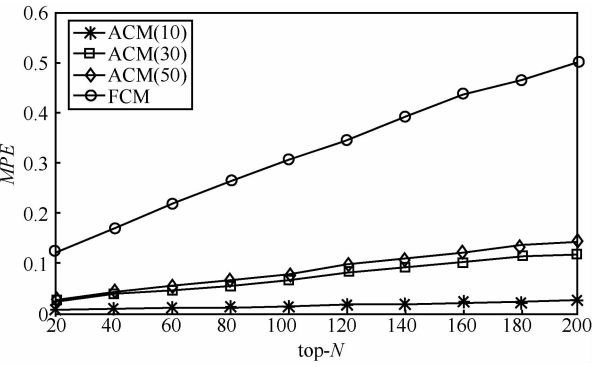
接着计算模型的吞吐计算能力,因为运行时间受分布式框架节点数据和数据传输影响较大,因此吞吐能力测试在单节点上进行。取 170 880 条数据,在单节点中测量 DCI 计算时间和 sketch cube 计算时间。DCI 时间表示计算数据单元标识集合的时间,sketch 时间表示计算存入二维数组的偏移值的时长。测试取 $w=1\ 021$ 。如图 9 所示,折线表示 DCI 集合的大小变化。从 DCI 运行时间和 DCI 集合变



(a) $w=521, t=10$



(b) $w=1\,021, t=10$



(c) $w=2\,027, t=10$

图 8 w 和 $ACM(m)$ 对 MPE 的增长趋势

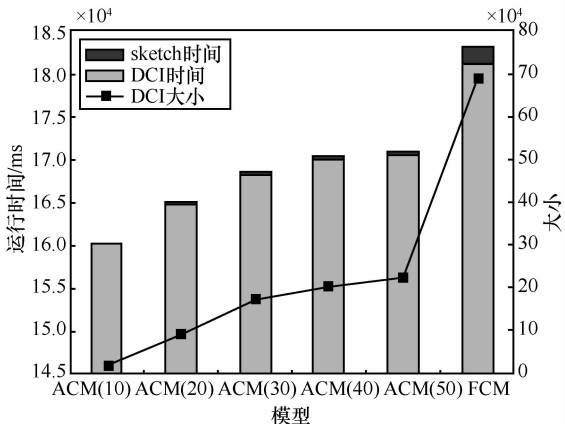


图 9 不同模型的运行时间

化曲线看,两者之间成正比关系,即模型的裁剪能力越强,DCI 集合越小,DCI 运行时间越短。对于任意模型,DCI 运行时间远大于 sketch 计算时间。这些结论将被用于 sketch cube 模型的实现中。

4.5 实时挖掘效率

上述实验过程把每条记录可能的维度组合保存在类线性空间中。本实验将统计数据存储于 MongoDB 中,使用应用 ID 和时间窗口对数据检索。通过时间序列索引模型可知 sketch cube 支持任意时间窗口的组合。实验取 3 个不同的应用做实时查询,其效率分析见表 5。

表 5 查询效率分析

时间窗口	数据集大小	运行时间/ms
10 min	7 035	67
20 min	13 470	72
30 min	19 815	81

每个应用分别以 10、20、30 min 为长度,随机选择高维度值产生数据单元,进行 OLAP 查询。数据大小表示查询所涉及应用的日志数据大小。运行时间包括用时间序列索引找到相关的存储单元,通过数据单元标识模型产生唯一自然数标识,使用最小计数模型找到该数据单元的估计值所需的时间总和。从运行时间看,因为数据已经做了索引和统计,时间片长短和数据量大小对查询时间变化较小。同时查询时间都在毫秒级,因此可以满足实时查询的需求。

4.6 空间效率

由于目前已有的对流数据的 OLAP 框架都是基于先存储数据,再进行分析的过程,和本文所述的实时统计框架有所不同。因此,本节主要理论分析数据在 HashMap 和 sketch cube 中的存储空间对比。假设,所有数据存储的基本单元都是整型,则 HashMap 的结构可以表示为 $HashMap<Integer,Integer>$,key 为维度计数器产生的数据单位标识,value 为数据单元的统计个数。因此,对于每个数据单元需要 2 个 Integer 大小空间存储,整体所需存储空间为 $2 \times DCI$ 个 Integer。而使用 sketch cube 的存储大小固定为 $w \times d$ 个 Integer。例如,在表 4 中可得,ACM(10)在 20 min 产生的 DCI 为 15 896,则 HashMap 的空间为 $15\,896 \times 2 = 31\,792$ 个 Integer,本实验中 sketch cube 大小为 $1\,021 \times 5 = 5\,105$ 个 Integer。因此,压缩率为 $5\,105 / 31\,792 = 0.16$ 。同理,可得 ACM(30)和 ACM(50)压缩率分别为 0.028 和 0.022。由此可得,随着 ACM 裁剪效率的下降,需要保存的数据单元数量增加,压缩率会明显降低,空间使用效率提高。



5 sketch cube 平台构建过程

5.1 系统结构

本节主要介绍使用 storm 开源流框架搭建 sketch cube 的过程。storm 框架使用拓扑来描述不同计算节点之间的关系,根据前文的模块定义,把 sketch cube 的不同模块功能映射到 storm 的 bolt 中,且需要根据计算节点之间的 I/O 吞吐率和 CPU 计算量,调整节点的并发度,以达到系统整体的平衡。系统部署在第 4.1 节描述的 4 台服务器中,其结构如图 10 所示。

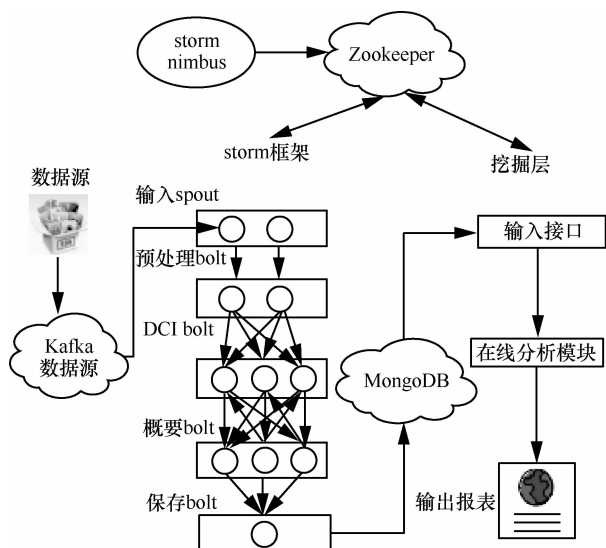


图 10 storm 实现 sketch cube 框架

系统分为 3 个部分,数据源负责收集相关多维度流数据信息,并保存在 Kafka 框架中;storm 框架使用 5 个不同的计算节点,分别为输入数据接收、字典化处理、DCI 计算,存储节点计算和存储节点。处理后的数据被保存在 MongoDB 数据库中。实时挖掘层(mining layer)从 MongoDB 中获得压缩后的数据单元统计值,并且进行相关在线分析模块(OLAM)操作得到结果返回给用户。

5.2 相关示例

对例 1 中的场景,给定应用名称和时间片段,实时统计不同的数据单元下的度量值,本实验使用 count 计算数据单元的大小,最后选择和该时段这个应用最相关的 top-50 个数据单元。输出所在数据单元的维度值。对于给定应用 app_i ,对于数据单元 c 的相关度(correlation)可以用式(21)所示:

$$correlation_c^{app_i} = \frac{m(app_i)}{\sum_{i=0}^n m(app_i)} \quad (21)$$

其中, m 表示数据单元度量值。

如对于流数据立方体 $c = ((Wi-Fi, hz, Samsung), 20140510081559)$, 包含应用 A 的条数为 200 条,应用 B 为 300 条,则 $correlation_c^{app_A} = 0.4$, $correlation_c^{app_B} = 0.6$ 。

对上述 top-50 的数据单元相关属性值进行统计,按每个属性值出现次数多少排序,并产生如图 11 所示的标签云图像。从图 11 可得,不同时间窗口下的标签云发生变化,这表明不同时间段的应用受众群体在发生变化。该类图像报表可以更好地帮助用户实时定位目标客户。

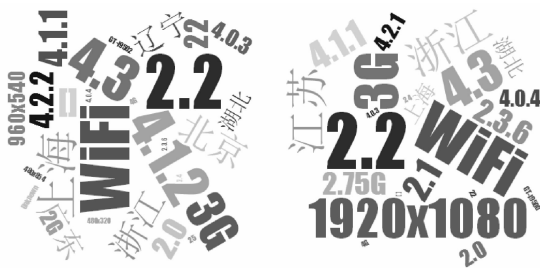


图 11 标签云推荐

6 总结和展望

本文研究一种对海量流数据进行 OLAM 操作的框架 sketch cube:

- 提出数据单元标识模型及其改进算法;
- 引入计数最小概要模型保存统计结果,并给出其适用场景;
- 提出基于上下限的数据单元裁剪方法,提高存储效率和准确率;
- 基于倾斜时间窗口的索引模型可快速计算任意时间片中的统计结果;
- 真实数据集实验中,比较了相关参数对模型的影响,体现模型的优越性和可用性。

sketch cube 只支持流数据分布型的聚合操作,在下一步工作中,将研究较为复杂的代数型和整体型聚合操作在流数据 sketch cube 模型中的应用。同时把计算过程中的性能瓶颈通过合理设计拓扑结构分散到不同机器节点中,以提高流数据的处理能力和稳定性。

参考文献

- 1 Aggarwal C C. An Introduction to Data Streams. Data Streams. Springer US, 2007

- 2 Hellerstein J M, Haas P J, Wang H J. Online aggregation. ACM SIGMOD Record, 1997, 26(2): 171~182
- 3 Zhang X, Chou P L, Dong G. Efficient computation of iceberg cubes by bounding aggregate functions. IEEE Transactions on Knowledge and Data Engineering, 2007, 19(7)
- 4 Chen Y, Dong G, Han J, *et al.* Multi-dimensional regression analysis of time-series data streams. Proceedings of the 28th International Conference on Very Large Data Bases, VLDB Endowment, Hong Kong, China, 2002: 323~334
- 5 胡文瑜, 孙志挥, 吴英杰. 数据挖掘取样方法研究. 计算机研究与发展, 2009, 48(1): 45~54
- 6 De Rougemont M, Cao P T. Approximate answers to OLAP queries on streaming data warehouses. Proceedings of the Fifteenth International Workshop on Data Warehousing and OLAP, Maui, HI, USA, 2012: 121~128
- 7 Babcock B, Shinath B, Mayur D, *et al.* Models and issues in data stream systems. Proceedings of the 21st ACM Symposium on Principles of Database Systems, Madison, Wisconsin, USA, 2002: 1~16
- 8 Chandrasekaran S, Cooper O, Deshpande A. TelegraphCQ: continuous dataflow processing for an uncertain world. Proceedings of the Conf on Innovative Data Systems Research, Asilomar, CA, USA, 2003
- 9 Hetal T, Nikolay L, Hamid M, *et al.* SMM: a data stream management system for knowledge discovery. Proceedings of International Conference on Data Engineering, Hannover, Germany, 2011: 757~768
- 10 Rosenberg A L. Efficient pairing functions-and why you should care. International Journal of Foundations of Computer Science, 2003, 14(1): 3~17
- 11 Zhang D, Zhai C, Han J, *et al.* Topic modeling for OLAP on multidimensional text database: topic cube and its applications. Stat Anal Data Min, 2009, 2(56): 378~395
- 12 Ding B, Zhao B, Lin C, *et al.* Topcells: keyword-based search of top-*k* aggregated documents in text cube. Proceedings of International Conference on Data Engineering (ICDE), Long Beach, USA, 2010: 381~384
- 13 Lin C, Ding B, Han J, *et al.* TextCube: computing measures for multidimensional text database analysis. Proceedings of the 8th IEEE International Conference on Data Mining (ICDM), 2008: 905~910
- 14 Liu X, Tang K Z, Hancock J, *et al.* A text cube approach to human, social and cultural behavior in the Twitter stream. LNCS 7812, 2013: 321~330
- 15 Cuzzocrea A. Retrieving accurate estimates to OLAP queries over uncertain and imprecise multidimensional data streams. Proceedings of the 23rd International Conference on SSDBM, Portland, OR, USA, 2011
- 16 Aggarwal C C. Managing and Mining Sensor Data. New York: Springer US, 2013
- 17 张进, 邬江兴, 刘勤让. 4种技术型 Bloom Filter 的性能分析与比较. 软件学报, 2010, 21(5): 1098~1114
- 18 Cormode G, Hadjieleftheriou M. Finding frequent items in data streams. Proceedings of the VLDB Endowment, 2008, 1(2): 1530~1541
- 19 Considine J, Hadjieleftheriou M, Li F, *et al.* Robust approximate aggregation in sensor data management systems. ACM Transactions on Database Systems (TODS), 2009, 34(1)
- 20 Han J, Kamber M, Pei J. Data Mining Concepts and Techniques. Elsevier Ltd, 2012
- 21 Cormode G, Muthukrishnan S. An improved data stream summary: the count-min sketch and its applications. J Algorithms, 2005, 55(1): 58~75
- 22 Cormode G, Muthukrishnan S. Summarizing and mining skewed data streams. Proceedings of SDM, Trondheim, Norway, 2005
- 23 Giannella C, Han J, Pei J, *et al.* Mining frequent patterns in data streams at multiple time granularities. Next Generation Data Mining, 2003(212): 191~212

[作者简介]



金苍宏,男,浙江大学博士生,主要研究方向为信息检索、数据挖掘、大数据框架。



刘泽民,男,浙江大学硕士生,主要研究方向为大数据平台、流数据处理、数据挖掘。



吴明晖,男,博士,浙江大学城市学院教授,主要研究方向为软件工程、人工智能。

应晶,男,博士,浙江大学教授、博士生导师,主要研究方向为软件开发方法、软件体系结构、软件工程规范等。

(收稿日期:2014-06-18)