滑动窗口规模的动态调整算法*

李建中[1,2+], 张冬冬[1]

[1](哈尔滨工业大学 计算机科学与技术学院,黑龙江 哈尔滨 150001)

[2](黑龙江大学 计算机科学与技术学院,黑龙江 哈尔滨 150080)

Algorithms for Dynamically Adjusting the Sizes of Sliding Windows

LI Jian-Zhong[1,2+], ZHANG Dong-Dong[1]

[1](College of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China)

[2](College of Computer Science and Technology, Heilongjiang University, Harbin 150080, China)

+ Corresponding author: Phn: +86-451-86415827, E-mail: lijzh@hit.edu.cn, http://www.hit.edu.cn

Received 2004-07-26; Accepted 2004-09-06

Li JZ, Zhang DD. Algorithms for dynamically adjusting the sizes of sliding windows. *Journal of Software*, 2004,15(12):1800~1814.

http://www.jos.org.cn/1000-9825/15/1800.htm

Abstract: The problem of dynamically adjusting the sizes of sliding windows when the rates of data streams or continuous queries change in data stream systems is studied in this paper. Based on the amount of available memory resource and the requirement of queries, three classes of algorithms for dynamically adjusting the sizes of sliding windows are proposed. These algorithms provide three levels of quality of service to all kinds of continuous queries and enhance the efficiency and effectiveness of processing continuous queries. Analytical and experimental results show that the algorithms can be applied to data stream systems effectively.

Key words: data stream; sliding window; continuous query; adjustment of sliding window size

摘 要: 讨论当数据流系统的数据流流速或连续查询发生变化时,滑动窗口规模的动态调整问题.根据可用内存空间大小和连续查询需求,提出了3类动态调整滑动窗口规模的算法,实现了对连续查询3种服务质量级别的支持,提高了连续查询处理的效率和效果.理论分析与实验结果表明,提出的算法可以有效地应用于数据流系统.

关键词: 数据流;滑动窗口;连续查询;滑动窗口规模调整

中图法分类号: TP391 **文献标识码**: A

* Supported by the National Natural Science Foundation of China under Grant No.60273082 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2002AA444110 (国家高技术研究发展计划(863)); the National Grand Fundamental Research 973 Program of China under Grant No.G1999032704 (国家重点基础研究发展规划(973)); the Natural Science Foundation of Heilongjiang Province of China under Grant No.zjg03-05 (黑龙江省自然科学基金)

作者简介: **李建中**(1950－),男,黑龙江哈尔滨人,博士,教授,博士生导师,主要研究领域为数据库,并行计算;**张冬冬**(1976－),男,博士生,主要研究领域为数据流处理,海量数据管理.

1 问题提出

近年来,许多应用领域中出现了大量的数据流.数据流不同于存储在磁盘上的关系数据,而是以流的形式快速、无限、连续、实时地出现.典型的数据流包括无线传感器网络应用环境中由传感器传回的各种监测数据、股票交易所的股票价格信息数据、网络监测系统与道路交通监测系统的监测数据、电信部门的通话记录数据,以及网站的日志信息等.能够处理数据流的系统环境统称为数据流系统.数据流可以被看作是一个允许元素重复出现的无限集合,集合中每个元素具有形式$\langle s,t\rangle$,其中 s 是数据流的一个数据项(可能是一个元组),t 为标识 s 的时间戳,t 的取值可由 s 进入数据流系统的时间或者数据源产生 s 的时间确定[1].由于内存资源有限,数据流无限,有限的内存空间无法存储数据流中的全部数据,因此,数据流系统通过在内存中开辟滑动窗口(sliding window)来保存最近一段时间内到达的数据流数据,实时地支持查询请求.与传统数据库系统的查询不同,数据流系统的多数查询在系统中长期处于执行状态.随着数据流源源不断地进入数据流系统,这些查询不断地接收新数据,进行查询处理,产生新的查询结果.这些查询称为连续查询(continuous query).数据流系统可以同时存在多个数据流和多个滑动窗口.不失一般性,我们假设一个数据流对应一个滑动窗口,同一个滑动窗口可以被多个不同的连续查询共享,同一个连续查询也可能查询多个滑动窗口中的数据.

目前的数据流连续查询技术主要考虑滑动窗口中的数据,并取得了一些研究成果[2~5].这些研究工作都基于如下假设:滑动窗口的大小固定不变,查询处理所涉及的数据(以下简称查询范围)限定在滑动窗口中的数据集合.然而,在很多实际应用中,这一假设是不成立的,原因是:

(1) 由于内存资源有限,滑动窗口的规模也是有限的,仅能存储无限数据流的一个子集合.在实际应用中,数据流上连续查询的查询范围是不可预知的,难以保证其仅涉及滑动窗口中的数据集合.

(2) 由于数据流上的不同连续查询的查询范围不同,而且数据流的流速随着时间不断地变化,因此,滑动窗口的规模需要实时地调整,以适应连续查询与数据流流速的动态变化.如果滑动窗口的规模固定不变,则会出现如下问题:某段时期内部分滑动窗口中有大量存储空间空闲,而另一部分滑动窗口则由于窗口太小难以支持连续查询的处理.

我们用一个例子来说明动态调整滑动窗口规模的必要性和重要性.假设数据流系统中存在两个滑动窗口 w_1 和 w_2,数据流的平均流速均为 1byte/s,可分配给滑动窗口的内存资源总和为 50bytes.两个滑动窗口规模的分配策略是:w_1 可以存储最近 25s 的数据,w_2 也可以存储最近 25s 的数据.设 q_1 是 w_1 上的连续查询,q_2 是在 w_2 上的连续查询,q_1 的查询范围是最近 20s 的数据,q_2 的查询范围是最近 15s 的数据.显然,目前 w_1 和 w_2 的规模可以支持 q_1 和 q_2 的处理.现在有一个新的连续查询 q_3 在 w_2 上开始执行(即 q_3 与 q_2 共享 w_2),其查询范围是最近 30s 的数据.如果固定 w_1 和 w_2 的大小不变,则由于分配给 w_2 的内存太小而不能支持 q_3.然而,w_1 具有能够存储 25−20=5s 数据流数据的空闲空间.在保持所有滑动窗口的内存资源总和不变的情况下,如果将 w_1 上空闲内存空间出让给 w_2 使用,即令 w_1 存储最近 20s 的数据,w_2 存储最近 30s 的数据,则两个滑动窗口可以支持所有的连续查询.

上例足以说明,当连续查询开始执行、结束运行或数据流流速发生变化时,数据流系统中采取固定滑动窗口规模的策略是不现实的,不能有效地支持所有连续查询.因此,我们需要推翻已有数据流处理技术中关于滑动窗口规模固定这一假设,寻求一种灵活地调整滑动窗口规模的机制,更有效地支持数据流系统中的连续查询.

本文根据连续查询的查询范围存在差异以及数据流流速动态变化等情况,提出了动态调整滑动窗口规模的算法,使得在内存资源有限的情况下,能够合理地调整每个滑动窗口的大小,最大程度地支持多连续查询,同时使连续查询的误差总和降到最低.本文的基本思想是根据连续查询的查询范围与系统能够提供给所有滑动窗口的内存资源总和之间的关系,现将滑动窗口规模的调整问题分为以下 3 种情况:

(1) 当滑动窗口的内存资源总和足够大时,采用预测方法调整滑动窗口的大小,目的是有效地支持已注册的和未来的连续查询.

(2) 当滑动窗口的内存资源总和不能保证所有的连续查询在任何时刻都可以输出零误差查询结果时,本文给出优化的调整算法,使得调整后的滑动窗口能够支持所有连续查询在任何时刻有能力输出在允许误差范围内的查询结果,并且使得查询误差总和降到最低.

(3) 当滑动窗口的内存资源总和不能保证所有的连续查询在任何时刻都可以输出在允许误差范围内的查询结果时,本文给出相应的优化调整算法和近似优化调整算法,使得调整后的滑动窗口能够支持所有连续查询在允许的延迟时间之内输出在允许误差范围内的查询结果.

本文给出了上述调整算法的理论分析结果与实验结果,充分说明了算法的有效性.

2 相关工作

文献[6,7]综述了数据流技术的相关研究成果以及热点研究问题.目前存在很多数据流项目,如 Stanford 大学的 STREAM[8],加州大学 Berkeley 分校的 Telegraph[9,10],布朗大学的 Aurora[11]等.STREAM 项目旨在研究通用的数据流管理系统,主要研究数据流查询语言的语法及语义[6],数据流系统中操作符的调度[12]与资源优化管理问题[8,13].Telegraph 项目的特点是能够自适应地查询处理数据流[14].Aurora 项目研究了数据元组在操作符之间的优化路由问题[11],能够更加充分地优化使用系统资源以提高服务质量(QoS)[11,15~17].文献[2~5]从各方面提高数据流查询的效率和精度,但是没有考虑优化利用数据流系统资源的问题.

文献[18]讨论了如何确定一个查询是否能够在数据流系统中有限的内存空间中执行,但是没有讨论近似查询与内存资源需求之间的关系.文献[12]考虑了优化调度查询计划中操作符的执行顺序,以节省操作符之间的缓存区资源,但是没有讨论系统输入缓存区的利用率问题.文献[5]通过优化调度共享滑动窗口的 Join 查询执行顺序来降低查询的平均相应时间,最大化查询的吞吐量,但是没有考虑共享滑动窗口规模的优化调整问题.文献[13]在分布式数据流系统中调整远端数据源上的谓词过滤条件,达到减少通信的目的,但是没有讨论内存资源的优化使用问题.总而言之,尽管上述研究工作涉及到了优化使用数据流系统资源的问题,但是这些研究成果都假定滑动窗口的规模固定不变,而没有考虑优化分配滑动窗口占用的内存资源问题.

文献[19]讨论了基于滑动窗口的无限数据流的 Join 查询代价模型,考虑了调整滑动窗口规模对查询效率的影响,但存在以下几个问题:(1) 仅通过实验对几种比较直观的调整策略进行了比较,而没有从理论上给出最优的调整策略;(2) 没有考虑多个查询共享滑动窗口时的优化调整问题;(3) 没有讨论近似查询对滑动窗口规模调整策略的影响.本文充分讨论并解决了上述关键性问题.

3 问题描述

为方便起见,我们把本文经常使用的符号及其定义列在表 1 中.

Table 1 Symbols used in this paper

表 1 本文中用到的符号

Symbol	Description	Symbol	Description
W	Set of sliding windows	Q	Set of continuous queries
$DSize(w)$	Size of each tuple of w	$Rate(w)$	Average rate of stream data entering w
M	Size of available memory	D	$d=\|W\|$, i.e., number of sliding windows
$W(w)$	Window-Width of w	$W_Y(w)$	Window-Width of w after adjustment
$Max\_T(w)$	Maximal query-width over w	$Min\_T(w)$	Minimal valid width of w
$Q_r(q)$	Querying-Range of q	$Q_e(q)$	Width mapped from tolerable error of q
$Q_d(q)$	Tolerable delay of q when obtaining result	$Q_{r\text{-}e}(q)$	Equal to $Q_r(q)-Q_e(q)$
$A(w)$	Free-Memory of w	$Min\_D(w)$	Minimal adjustment
$B(w)$	Gain-Memory of w	$P(w,q)$	If $U(w,q)=1$ and $W(w)<Q_r(q)$, then 1; otherwise, 0
$U(w,q)$	If q is over w, then 1; otherwise,0	$G(w,x)$	Adjustment-Gain of w
$E(w)$	Accumulate-Error of w	$Exch(w)$	Exchange-Memory of w
$T_P(w)$	Adjusting-Period of w	$Share(Z)$	Sharing-Memory of serial-adjusting-group Z
Z	Serial-Adjusting-Group	$P_T(Z)$	Cyclic-Period of serial-adjusting-group Z

3.1 滑动窗口

设数据流按照时间戳的先后顺序进入滑动窗口.任意时刻每个滑动窗口中的数据可以表示成序列

$$\langle s_l,t_l\rangle,\langle s_{l+1},t_{l+1}\rangle,\ldots,\langle s_{u-1},t_{u-1}\rangle,\langle s_u,t_u\rangle \quad (1)$$

其中,对于 $l\le i\le j\le u, t_i\le t_j$.滑动窗口分为两类[6],一类是基于元组个数定义的滑动窗口,即窗口内保存最近到来的 K

个元组,也即在任意时刻序列(1)满足条件 $u-l=K$.另一类是基于时间定义的滑动窗口,即存储最近 T 时间内到达的元组,亦即在任意时刻序列(1)满足条件 $t_u-t_l=T$.本文将重点讨论基于时间定义的滑动窗口.本文的算法完全可以扩展应用于基于元组个数定义的滑动窗口类型.

为方便描述,我们使用如下定义的滑动窗口的宽度来描述基于时间定义的滑动窗口的大小或规模.

定义 1(宽度). $\forall w\in W$,滑动窗口 w 中的数据形如序列(1)所示,t_u-t_l 定义为 w 的宽度,记作 $W(w)$.

调整滑动窗口规模就是调整滑动窗口的宽度.设任意滑动窗口 w 中每个数据的大小为 $DSize(w)$,流入 w 的数据流的平均流速为 $Rate(w)$,则 w 占用的内存空间为 $W(w)\times DSize(w)\times Rate(w)$.设数据流系统能够分配给所有滑动窗口的内存资源总和(以下简称可用内存空间)为 M,任何一个滑动窗口动态调整算法必须满足约束条件:$M\geq\Sigma_{w\in W}W(w)\times DSize(w)\times Rate(w)$.

3.2 连续查询

下边是一个典型的类似于 SQL 语句的连续查询语句[1,7]:

SELECT AVG(*price*)
FROM sliding_window AS *w* [RANGE *Now*−100, *Now*]
WHERE *predicate*
ERROR (2%)
EVERY (5)
DURATION [70,500]

其中,FROM 子句说明了查询对象(如滑动窗口 w)和查询范围(如从时刻 $Now-100$~当前时刻 Now),WHERE 从句给出了查询条件 *predicate*,ERROR 子句说明查询结果允许的相对误差范围(如 2%),EVERY 子句定义了相邻两次返回查询结果的最大时间间隔或允许的延迟时间(记为 $Q_d(q)$,如 $Q_d(q)$=5 个时间单位),DURATION 子句说明该查询执行的时间区间(如从时刻 70~时刻 500).

对于上述查询,滑动窗口 w 需至少存储从 $Now-100$~Now 时刻的数据.我们称 $Now-(Now-100)=100$ 为这个连续查询的查询宽度,记为 $Q_l(q)$.查询宽度表征了滑动窗口必须至少具有大小为 $Q_l(q)$的宽度才能保证查询可以得到零误差度的查询结果.

本文通过一个函数映射 f_q 把查询 q 的允许误差范围值映射成一个时间跨度值(记为 $Q_e(q)$),使得滑动窗口的宽度至少为 $Q_l(q)-Q_e(q)$时才能保证查询可以得到允许误差范围内的查询结果.我们可以采用 Load Shedding 技术[15,22]和抽样技术[20,21]构造映射函数 f_q.

在下边的讨论中,我们用 $U(w,q)$=1 表示查询 q 是滑动窗口 w 上的查询,$U(w,q)$=0 表示查询 q 不是滑动窗口 w 上的查询.

滑动窗口对连续查询支持的质量受宽度的限制.根据宽度与连续查询的特征,我们可以将滑动窗口对连续查询支持的质量分为 3 个级别.

定义 2(A 级支持,B 级支持,C 级支持). 对于任意滑动窗口 w,设 $W(w)^{(i)}$是 w 在 i 时刻的宽度,q 是 w 上的一个连续查询.对于任意时刻 i,若 $W(w)^{(i)}\geq Q_l(q)$,则称 w 以 A 级支持连续查询 q;若 $W(w)^{(i)}\geq Q_l(q)-Q_e(q)$,则称 w 以 B 级支持连续查询 q.如果存在时刻序列 $t_1<t_2<t_3<\ldots$,且$\forall k,t_{k+1}-t_k\leq Q_d(q)$和 $W(w)^{(t_k)}\geq Q_l(q)-Q_e(q)$同时成立,则称 w 以 C 级支持连续查询 q.

A 级支持表示滑动窗口可以支持连续查询在任何时刻输出零误差的查询结果.B 级支持表示滑动窗口可以支持连续查询在任何时刻输出在允许误差范围内的查询结果.C 级支持表示滑动窗口可以支持连续查询在允许的延迟时间间隔内输出在允许误差范围内的查询结果.显然,若滑动窗口 w 能以 A 级支持连续查询 q,则 w 必能以 B 级支持或 C 级支持 q;同理,若 w 能以 B 级支持 q,则 w 必能以 C 级支持 q.

3.3 问题的分类

我们可以从任意一个查询 q 中提取出如下信息:q 的 ID(Qid)、q 涉及的窗口 ID(Wid)、q 的起始和终止执行时间(Begin 和 End)等,见表 2.滑动窗口的状态信息见表 3,包括滑动窗口 ID(wid)、每个数据的大小($DSize(w)$)、

数据流的平均流速($Rate(w)$)、当前时刻的宽度($W(w)$)和访问滑动窗口 w 的所有连续查询中最大的查询宽度($Max\_T(w)$).

Table 2 Continuous-Query-Information table
表 2 连续查询信息表的内容

Qid	Wid	$Q_t(q)$	$Q_e(q)$	$Q_d(q)$	Begin	End
q_1	w_1	80	8	5	3	200
q_2	w_2	110	5	2	0	200

Table 3 Sliding-Window-Information table
表 3 滑动窗口信息表的内容

Wid	$Max\_T(w)$	$DSize(w)$	$Rate(w)$	$W(w)$
w_1	100	100	2	100
w_2	110	80	4	120

连续查询和数据流流速的动态变化情况都会反映到连续查询信息表和滑动窗口信息表的内容变化上.本文研究的滑动窗口规模的动态调整问题就是根据连续查询信息表和滑动窗口信息表内容的变化,动态调整滑动窗口的宽度,使得在给定可用内存空间大小约束下能够支持所有的连续查询,并且保证最小化查询结果的误差总和或者最大化内存的利用率.

定义 3(空闲空间). $\forall w\in W$,定义$(W(w)-Max_T(w))\times DSize(w)\times Rate(w)$为滑动窗口 w 的空闲空间,记为 $A(w)$.

窗口空闲空间表示当滑动窗口 w 以 A 级支持所有涉及 w 的连续查询时,能够空闲出的最大可用内存空间(如图 1 中滑动窗口 w_2 的阴影区域面积所示).

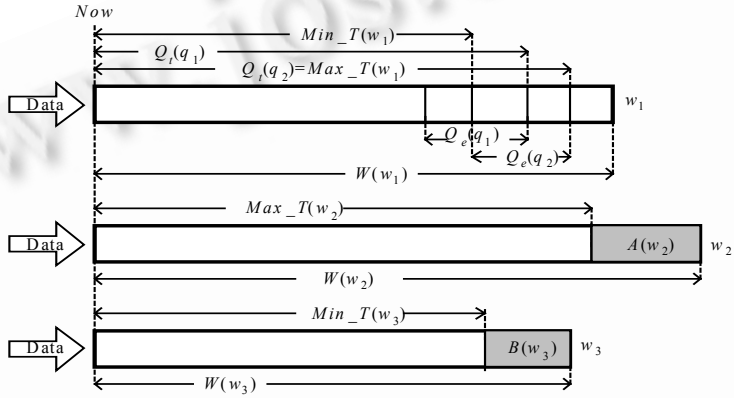

Fig.1 Example of relationship between sliding windows and continuous queries

图 1 滑动窗口及连续查询的示例

定义 4(最小有效宽度). $\forall w\in W$,定义 $\max\{Q_{t\text{-}e}(q)|q\in Q\wedge U(w,q)=1$,其中 $Q_{t\text{-}e}(q)=Q_t(q)-Q_e(q)\}$为滑动窗口 w 的最小有效宽度,记为 $Min\_T(w)$.

滑动窗口的最小有效宽度表示当滑动窗口 w 以 B 级支持所有涉及 w 的连续查询时,w 必须具有的最小宽度.

定义 5(盈余空间). $\forall w\in W$,定义$(W(w)-Min_T(w))\times DSize(w)\times Rate(w)$为滑动窗口 w 的盈余空间,记为 $B(w)$.

窗口盈余空间表示当滑动窗口 w 以 B 级支持所有涉及 w 的连续查询时,能够空闲出的最大可用内存空间(如图 1 中滑动窗口 w_3 的阴影区域面积).

滑动窗口规模的动态调整问题分为以下 3 类:

(1) 当$\Sigma_{w\in W}\ A(w)\geq 0$ 时,在可用内存空间大小约束下所有滑动窗口都能够以 A 级支持所有连续查询.因此,滑动窗口规模的调整任务是根据已注册的连续查询的特征,采用预估测方法,提前调整滑动窗口的宽度,以更好地支持未来连续查询.第 4.1 节将给出相应的调整算法,称为 A 级调整算法.

(2) 当$\Sigma_{w\in W}\ A(w)<0\leq\Sigma_{w\in W}\ B(w)$时,在可用内存空间大小约束下所有滑动窗口都能以 B 级支持所有连续查询,但是不能以 A 级支持全部连续查询.因此,滑动窗口规模的调整目标是使全部连续查询产生的误差总和降到最低.第 4.2 节将给出相应的调整算法,称为 B 级调整算法.

(3) 当$\Sigma_{w\in W}\ B(w)<0$ 时,在可用内存空间大小约束下所有滑动窗口都不能以 B 级支持所有连续查询.因此,滑动窗口的调整目标是使用最小可用内存空间实现全部连续查询的 C 级支持.第 4.3 节将给出相应调整算法,

称为 C 级调整算法.

4 滑动窗口规模的调整算法

本节仅介绍调整滑动窗口规模的 3 种策略及算法.

4.1 A级调整算法

当$\Sigma_{w\in W}A(w)\geq 0$时,即当所有滑动窗口的空闲空间总和大于等于 0 时,我们需要在确保 A 级支持所有连续查询的条件下,通过分析连续查询的特性,采用预估测方法,为部分滑动窗口分配较多的可用内存空间以支持更多的未来连续查询.一个直观的想法是:$\forall u,v\in W$,若 $Max_T(u)>Max_T(v)$成立,则说明 u 上的连续查询的查询宽度大于 v 上连续查询的查询宽度,亦即 u 上未来查询的查询宽度大于 v 上未来查询的查询宽度的概率较大.在这种情况下,u 分得的可用内存空间应该大于 v 分得的可用内存空间.基于上述思想,滑动窗口规模调整的策略是:按照各个滑动窗口上现有的最大查询宽度的比例调整各个滑动窗口的宽度.下面是相应的算法描述:

输入: M 以及$\Sigma_{w\in W}A(w)\geq 0$.

输出: $\forall w\in W$,求调整后的宽度 $W_Y(w)$的大小,使得$\Sigma_{w\in W}W_Y(w)\cdot DSize(w)\cdot Rate(w)\leq M$ 且 $W_Y(w)\geq Max\_T(w)$.

A-Support( )

(1) $\forall w\in W$,令 $W_Y(w)=Max\_T(w)$; /* 确保能够对访问 w 的连续查询实现 A 级支持 */

(2) 得到剩余可用内存空间 $M_R=M-\Sigma_{w\in W}Max\_T(w)\times DSize(w)\times Rate(w)$,即 $M_R\approx\Sigma_{w\in W}A(w)$;

(3) $\forall w\in W$,令 $W_Y(w)=W_Y(w)+(M_R\times Max\_T(w)/\Sigma_{k\in W}Max\_T(w))/(DSize(w)\times Rate(w))$.

此算法的时间复杂性为 $O(d)$,其中 d 为滑动窗口的个数.算法第(1)步保证了能以 A 级支持所有连续查询,第(2)、第(3)步体现了调整策略的基本思想.

4.2 B级调整算法

4.2.1 问题定义

定义 6(累计误差). $\forall w\in W$,定义$\Sigma_{q\in Q}P(w,q)\times(Q_t(q)-W(w))$为滑动窗口 w 的累计误差,记为 $E(w)$,其中当 $U(w,q)=1$ 且 $W(w)<Q_t(q)$时,$P(w,q)=1$,否则 $P(w,q)=0$.

本文采用累计误差来度量连续查询输出查询结果时产生的误差大小.当$\Sigma_{w\in W}A(w)<0\leq\Sigma_{w\in W}B(w)$时,在可用内存空间大小的约束下不能实现 A 级支持所有的连续查询.此时,滑动窗口规模的调整目标是:

(1) 实现对所有连续查询的 B 级支持;

(2) 使所有滑动窗口的累计误差总和降到最低,即$\Sigma_{w\in W}E(w)$最小.

根据以上调整目标,滑动窗口规模的调整步骤分两阶段进行.第 1 阶段把每个滑动窗口 w 的宽度设置为最小有效宽度,即 $W_Y(w)=Min\_T(w)$.第 2 阶段逐步将剩余的可用内存空间分配给各个滑动窗口(即逐步增大每个滑动窗口的宽度 $W_Y(w)$),达到最小化$\Sigma_{w\in W}E(w)$的目的.第 1 阶段比较简单,第 2 阶段比较复杂.下边我们详细介绍第 2 阶段的思想和方法.

第 2 阶段由多步组成,每步都使用贪心方法为一个滑动窗口增加新的可用内存空间,直至可用内存空间全部用尽.每一步调整之后都得到一个集合$\{s_w|w\in W$ 且 s_w 是 w 的宽度$\}$,称为规模调整的一个瞬时描述,记为 S.S 中滑动窗口 w 的宽度值表示为 $W^{(S)}(w)$.我们可以使用一个三元组$(w,W^{(S)}(w),\Delta x_w)$表示在瞬时描述 S 的基础上进行下一步调整时,滑动窗口 w 的宽度被增加了Δx_w.$(w,W^{(S)}(w),\Delta x_w)$称为 S 的一个调整活动(记作 I),$\Delta x_w>0$ 称为调整增量.

定义 7(宽度增量). 设 S_i 和 S_j 分别是第 i 和第 j 步滑动窗口规模调整后的瞬时描述,$i<j$.集合$\{x_w|w\in W$ 且 $x_w=W^{(S_j)}(w)-W^{(S_i)}(w)\}$为宽度增量,记为$\Delta X$,$\Delta X$ 中与滑动窗口 w 对应的元素表示为 $W^{(\Delta X)}(w)$.

假设有瞬时描述 S,S 的调整活动 $I=(w,W^{(S)}(w),\Delta x)$,以及宽度增量$\Delta X$,我们用 $S+I$ 表示将滑动窗口 w 的宽度由原来的 $W^{(S)}(w)$增加到 $W^{(S)}(w)+\Delta x$ 后形成的瞬时描述.我们用 $S-I$ 表示将滑动窗口 w 的宽度由 $W^{(S)}(w)$减少到 $W^{(S)}(w)-\Delta x$ 后形成的瞬时描述.同理,我们用 $S+\Delta X$ 表示$\{W^{(S)}(w)+W^{(\Delta X)}(w)|w\in W\}$,即将所有滑动窗口的宽度由 $W^{(S)}(w)$增加到 $W^{(S)}(w)+W^{(\Delta X)}(w)$后形成的瞬时描述.以下,$D(S,\Delta X)$表示由 S 到 $S+\Delta X$ 的变化过程中所有滑动窗口

减少的累计误差总和,S 称为ΔX 的基.$\Delta X+I$ 表示将ΔX 中滑动窗口 w 的宽度增量由 $W^{(\Delta X)}(w)$变化到 $W^{(\Delta X)}(w)+\Delta x$.

由以上分析可知,滑动窗口规模调整问题可以等价地定义为如下优化问题:

输入:初始瞬时描述 $S_0=\{Min\_T(w)|w\in W\}$.

输出:宽度增量$\Delta X=\{\Delta x_w|w\in W\}$,$\Delta X$ 满足:

(1) $D(S_0,\Delta X)=\max\{D(S_0,\Delta A)|\Delta A$ 是任意一个宽度增量$\}$,

(2) $\Sigma_{w\in W}(W^{(S_0)}(w)+W^{(\Delta X)}(w))\times DSize(w)\times Rate(w)\le M$. (2)

4.2.2 贪心调整算法

定义 8(调整收益). $\forall w\in W$,定义$|N(w,y)|/(DSize(w)\times Rate(w))$为滑动窗口 w 具有宽度 y 时的调整收益,记为 $G(w,y)$,其中 $N(w,y)=\{q|U(w,q)=1$ 且 $Q_t(q)>y\}$.

调整收益表征了滑动窗口 w 在具有宽度 y 时,把单位可用内存空间增加到 w 上会引发的累计误差总和的减少量.由于连续查询的查询宽度是离散值,滑动窗口的调整收益是递减阶梯函数.$\forall w\in W$,将 $G(w,y)$在定义域 $[Min_T(w),Max_T(w)]$上使调整收益值发生改变的有序点(简称收益点)列 $a_{w_1},\ldots,a_{w_i},\ldots,a_{w_{n_w}}$ 称为 w 的收益点序列,记为 L_w,其中 $1\le i<j\le n_w, a_{w_i}<a_{w_j}, a_{w_1}=Min\_T(w), a_{w_{n_w}}=Max\_T(w)$.$\forall w\in W$,$L_w$ 中的每个点确定了一个连续查询的查询宽度.设 a 和 b 是任意两个相邻收益点.由定义 8 可知,$G(w,y)$在[a,b)之间的取值处处相等,而且当 $y_i\le y_j$ 时,$G(w,y_i)\ge G(w,y_j)$.

定义 9(最优滑动窗口和最优调整活动). 设 S 是一个瞬时描述.满足 $G(u,W^{(S)}(u))=\max\{G(w,W^{(S)}(w))|w\in W\}$ 的滑动窗口 u 称为 S 的最优滑动窗口.L_u 中一定存在两个相邻收益点 a_{u_i} 和 $a_{u_{i+1}}$ 使得 $a_{u_i}\le W^{(S)}(u)<a_{u_{i+1}}$ 成立,相应的调整活动 $I=(u,W^{(S)}(u),\Delta x_u)$称为 S 的最优调整活动,其中$\Delta x_u=a_{u\,i+1}-W^{(S)}(u)$,称为 S 的最优增量.

定义 10(最优调整活动的覆盖). 设瞬时描述 S 的最优滑动窗口为 u,S 的最优调整活动 $I=(u,W^{(S)}(u),b)$.如果以 S 为基的宽度增量ΔX 满足下列条件之一:

(1) $W^{(\Delta X)}(u)\ge b$

(2) $\forall v\in W, v\ne u, W^{(\Delta X)}(v)=0$,且 $W^{(\Delta X)}(u)\ge 0$

则称ΔX 覆盖 I.

我们可以使用贪心算法求解问题(2).设 S 是任何一次调整滑动窗口规模时刻的瞬时描述,贪心算法进行如下调整:选择 S 的最优滑动窗口 u,执行 S 的最优调整活动,将 u 的宽度增加 S 的最优增量.我们通过 3 个引理证明问题(2)具有优化子结构和贪心选择性.

引理 1. 设滑动窗口 u 为初始瞬时描述 S_0 的最优滑动窗口,且 S_0 的最优调整活动为 $I_0=(u,W^{(S_0)}(u),b)$,宽度增量 A 为问题(2)的以 S_0 为基的某个优化解,则 A 覆盖了 I_0.

证明:如果 $W^{(A)}(u)\ge b$,则由定义 10 的条件(1),A 覆盖了 I_0.如果 $W^{(A)}(u)<b$,若$\forall v\in W, v\ne u, W^{(A)}(v)=0$,则由定义 10 的条件(2),$A$ 覆盖了 I_0.如果 $W^{(A)}(u)<b$,且"$\forall v\in W, v\ne u, W^{(A)}(v)=0$"不成立,则必$\exists v\in W, v\ne u, W^{(A)}(v)>0$.构造调整活动 $I_v=(v,W^{(S_0+A)}(v)-\Delta x_v,\Delta x_v)$ 和 $I_u=(u,W^{(S_0+A)}(u),\Delta x_u),\Delta x_u$ 和 Δx_v 满足:$\Delta x_v<W^{(A)}(v)$ 且 $\Delta x_u\times DSize(u)\times Rate(u)=\Delta x_v\times DSize(v)\times Rate(v)=\Delta M>0$.

令 $B=A-I_v+I_u$.B 也是一个以 S_0 为基的宽度增量.

$D(S_0,B)=D(S_0,A)-\Delta M\times G(v,W^{(S_0+A)}(v)-\Delta x_v)+\Delta M\times G(u,W^{(S_0+A)}(u))=D(S_0,A)+\Delta M\times(G(u,W^{(S_0+A)}(u))-G(v,W^{(S_0+A)}(v)-\Delta x_v))$.

显然,$W^{(S_0+A)}(v)-\Delta x_v=W^{(S_0)}(v)+W^{(A)}(v)-\Delta x_v>W^{(S_0)}(v)$.

由于滑动窗口 u 为 S_0 的最优滑动窗口且 $W^{(A)}(u)<b$,因此 $G(u,W^{(S_0+A)}(u))=G(u,W^{(S_0)}(u))$.由定义 8 可知,$G(v,W^{(S_0+A)}(v)-\Delta x_v)\le G(v,W^{(S_0)}(v))\le G(u,W^{(S_0)}(u))=G(u,W^{(S_0+A)}(u))$成立,从而可得 $D(S_0,B)\ge D(S_0,A)$.于是 B 也是一个以 S_0 为基的问题(2)的优化解,且 $W^{(B)}(u)>W^{(A)}(u)$.如果 B 仍没有覆盖 I_0,则以 B 为新的出发点,不断重复上述过程,最终可以构造出一个优化解覆盖 I_0. □

引理 2. 设滑动窗口 u 为初始瞬时描述 S_0 的最优滑动窗口,$I_0=(u,W^{(S_0)}(u),b)$是 S_0 的最优调整活动,宽度增量 A 是以 S_0 为基的问题(2)的某个优化解,且 A 覆盖 I_0.宽度增量 $A'=A-I_0$ 是问题(2)以 $S_1=S_0+I_0$ 为基的优化解.

证明:显然,A'是问题(2)以 S_1 为基的一个解,下面仅需证明 A'是优化解.若不然,设存在一个以 S_1 为基的问题

(2)的优化解 B',$D(S_1,B')>D(S_1,A')$.令宽度增量 $B=B'+I_0$,由于 I_0 是 S_0 的最优调整活动,则 B 是一个以 S_0 为基的问题(2)的优化解.由于 $A=A'+I_0$,$D(S_0,B)=D(S_0,I_0)+D(S_1,B')>D(S_0,I_0)+D(S_1,A')=D(S_0,A)$,与 A 是以 S_0 为基的问题(2)的优化解矛盾,因此 A'是问题(2)以 $S_1=S_0+I_0$ 为基的优化解. □

引理 1 和引理 2 证明了问题(2)具有优化子结构.下边的引理 3 证明了问题(2)具有贪心选择性.

引理 3. $\forall i\geq 0$,设瞬时描述 S_i 的最优调整活动为 I_i,且 $S_{i+1}=S_i+I_i$.若宽度增量 A 是以 S_0 为基的优化解,且覆盖 I_0,则存在正整数 k,使得 $A=\sum_{j=0}^{k} I_j$.

证明:对 k 作归纳法.若 $k=0$,由引理 1 可知,命题成立.设 $k<n$ 时命题成立,则当 $k=n$ 时,由引理 2 知,$A=A'+I_0$,A'是以 $S_1=S_0+I_0$ 为基的优化解.由归纳假设, $A'=\sum_{j=0}^{k} I_j$,于是 $A=\sum_{j=0}^{k} I_j$. □

根据引理 1~引理 3,我们可以给出如下的贪心调整算法,实现对连续查询的 B 级支持.

输入:M 以及 $\Sigma_{w\in W} Min\_T(w)\times DSize(w)\times Rate(w)\leq M<\sum_{w\in W} Max\_T(w)\times DSize(w)\times Rate(w)$.

输出:$\forall w\in W$,求调整后的宽度 $W_Y(w)$的大小.

B-Support( )

(1) $\forall w\in W$,令 $W_Y(w)=Min\_T(w)$; /* 确保能够实现 B 级支持所有连续查询 */

(2) 计算得到剩余可用内存空间 $M_R=M-\Sigma_{w\in W} Min\_T(w)\times DSize(w)\times Rate(w)$;

(3) $\forall w\in W$,产生滑动窗口 w 在区间[$Min\_T(w)$,$Max\_T(w)$]上的收益点序列 L_w;

(4) 令初始瞬时描述 $S_0=\{W_Y(w)|w\in W\}$,初始宽度增量$\Delta X=\{x_w=0|w\in W\}$,$i=0$;

(5) while ($M_R>0$)

(6) 找到瞬时描述 S_i 的最优滑动窗口 u,最优调整活动 $I_i=(u,W^{(S_i)}(u),b_i)$;

(7) if ($M_R\geq b_i\times DSize(u)\times Rate(u)$) $\Delta X=\Delta X+I_i$;

(8) else $W^{(\Delta X)}(u)=W^{(\Delta X)}(u)+M_R/(DSize(u)\times Rate(u))$;

(9) $M_R=M_R-b_i\times DSize(u)\times Rate(u)$,$S_{i+1}=S_i+I_i$,$i=i+1$;

(10) $\forall w\in W$,令 $W_Y(w)=W_Y(w)+W^{(\Delta X)}(w)$; /* 最终输出每个滑动窗口的最优宽度 */

上述算法描述中,第(1)步实现了第 1 阶段的调整.第(4)~第(9)步是问题(2)的贪心求解过程.第(10)步最终输出每个滑动窗口的最优宽度.第(3)步的时间复杂性为 $O(c_q\log c_q)$.第(5)~第(9)步最多需要执行 c_q 次.第(6)步的时间复杂性为 $O(d)$.于是,算法的总时间复杂性为 $O(\max\{d\cdot c_q,c_q\log c_q\})$,其中 c_q 表示系统中连续查询的总数,d 表示滑动窗口的个数.

定理 1. 当 $\Sigma_{w\in W} A(w)<0\leq\Sigma_{w\in W} B(w)$时,B-Support 算法能够产生滑动窗口规模调整问题的最优解.

证明:B-Support 算法第(4)~第(9)步按照引理 3 的贪心选择性进行局部优化选择,能够产生问题(2)的最优解.因此,B-Support 算法能够产生最优解. □

4.3 C级调整算法

4.3.1 问题的定义与分析

当 $\Sigma_{w\in W} B(w)<0$ 时,A-Support 和 B-Support 算法皆无效,无法对所有连续查询实现 A 级或 B 级支持.在这种情况下,我们可以尝试对所有连续查询实现 C 级支持.滑动窗口上连续查询一般都允许产生相邻的两次查询结果之间具有一段时间间隔.我们可以在一个连续查询的间歇期缩短与其相关的滑动窗口的宽度,释放部分可用内存资源供其他滑动窗口使用,在其产生查询结果时增大相关滑动窗口的宽度(申请部分可用内存资源),从而在有限可用内存空间大小约束下最大程度地实现对连续查询的 C 级支持.

一个滑动窗口可能同时被多个连续查询访问,而不同连续查询的执行周期也不同.我们可以从访问同一个滑动窗口的连续查询中选中一个连续查询(称为基准查询)作为参照系来动态调整滑动窗口的宽度,使得基准查询能够得到 C 级支持,而其他任何连续查询可以得到 B 级支持.于是,任何时刻滑动窗口 w 的内存资源可以分为两部分.一部分内存资源,由 w 固定占有,称为静态容量,保证了访问 w 的所有连续查询(除基准查询之外)能够得

到 B 级支持.另一部分内存资源与其他滑动窗口以轮转方式共享使用,称为交换容量,保证访问 w 的基准查询能够得到 C 级支持.

定义 11. 对于任意滑动窗口 w,w 上必有一个连续查询ξ,使得 $Q_{t-e}(\xi)=Q_t(\xi)-Q_e(\xi)=Min\_T(w)$.我们称$\xi$为 w 的基准查询,$Q_d(\xi)$称为 w 的调整周期,记为 $T_P(w)$.如果 w 上存在一个连续查询 q,使得 $Q_{t-e}(q)=\max\{Q_{t-e}(r)|r\in Q\wedge r\neq\xi\wedge U(w,r)=1\}$,则 $\min\{Min_T(w)-Q_{t-e}(q),T_P(w)\}$定义为 w 的最小调整量,记为 $Min\_D(w)$,否则 w 的最小调整量 $Min_D(w)$定义为 $T_P(w).Q_{t-e}(q)\times DSize(w)\times Rate(w)$的大小称为静态容量,记为 $Static(w).Min_D(w)\times DSize(w)\times Rate(w)$称为 w 的交换容量,记作 $Exch(w)$.

为了保证 C 级支持基准查询,在调整周期内滑动窗口的宽度变化量不能低于最小调整量,即滑动窗口占用的可用内存空间变化量不能小于交换容量.

如果多个滑动窗口同时申请内存资源,则有限的可用内存资源会造成部分滑动窗口饥饿.理想的情况是,所有的滑动窗口能够以串行方式申请可用内存资源.然而,由于受到滑动窗口的调整周期和最小调整量的约束,必然存在部分滑动窗口需要同时申请可用内存资源.为此,我们把滑动窗口分为多组,每组称为一个串行调整组.串行调整组内的滑动窗口按照一定的周期(称为串行调整周期),以串行方式申请和释放可用内存资源,使得在有限的可用内存空间条件下实现 C 级支持所有连续查询.各组之间的滑动窗口以并行方式申请和释放可用内存资源.

定义 12. 如果存在一个滑动窗口集合 Z,使得 $\Sigma_{w\in Z}Min\_D(w)\leq\min\{T_P(w)|w\in Z\}$,而且在任意时刻,$\forall w\in Z,W_Y(w)\geq Min\_T(w)-Min\_D(w)$,则称 Z 是一个串行调整组,称 $\max\{Exch(w)|w\in Z\}$为 Z 的共享内存量,记为 $Share(Z)$,称 $\min\{T_P(w)|w\in Z\}$为串行调整周期,记为 $P_T(Z)$.

一个滑动窗口集合 Z 可以构成串行调整组的条件是:(1) Z 中的每个滑动窗口都能够按照串行调整周期申请和释放可用内存资源;(2) Z 中的滑动窗口能够 C 级支持连续查询;(3) 任何时刻 Z 中的滑动窗口动态申请或释放可用内存资源的上界为共享内存量.由定义 11 和定义 12 可知,对于任意的滑动窗口 w,$\{w\}$是一个串行调整组.于是,为了实现 C 级支持连续查询,所有滑动窗口在任何时刻占用的可用内存空间总和最多为 $\Sigma_{w\in W}Static(w)+\Sigma_{Z\in C}Share(Z)$,其中 C 为 W 的串行调整组集合.

综上所述,当 $\Sigma_{w\in W}B(w)<0$ 时,滑动窗口规模的调整目标是使用最小的可用内存空间实现 C 级支持所有连续查询.具体思想是,寻找一种最优策略对滑动窗口进行分组,形成多个串行调整组,在保证每组内的所有连续查询都可以得到 C 级支持的前提下,最小化滑动窗口占用的可用内存空间总和.设该优化策略形成的串行调整组集合为 C,则所有滑动窗口占用的内存空间总和为 $M'=\Sigma_{G\in C}Share(G)+\Sigma_{w\in W}Static(w)$.于是,当 $\Sigma_{w\in W}B(w)<0$ 时,滑动窗口规模的调整问题可以定义为

输入:集合 W,对于$\forall w\in W,Min_D(w),T_P(w)$.

输出:子集族 C,且 $C\subseteq 2^W$,满足:

(1) $\Sigma_{G\in C}Share(G)+\Sigma_{w\in W}Static(w)=\min\{\Sigma_{G\in X}Share(G)+\Sigma_{w\in W}Static(w)|\forall X\subseteq 2^W\}$
$=\min\{\Sigma_{G\in X}Share(G)|\forall X\subseteq 2^W\}+\Sigma_{w\in W}Static(w)$;

(2) $\forall G\in C,\Sigma_{w\in G}Min\_D(w)\leq\min\{T_P(w)|w\in G\}$,即 G 是一个串行调整组;

(3) C 是 W 的一个完全划分. (3)

设问题(3)的最优解为 C^*,最少需要的共享内存量总和为 M^*,进一步,令 $Exch(w)=1$ 对$\forall w\in W$ 成立,则可知 $|C^*|=M^*$.于是,我们得到问题(3)的一个特例:

输入: 集合 W,对于$\forall w\in W,Min_D(w),T_P(w)$.

输出: 子集族 C,且 $C\subseteq 2^W$,满足:

(1) 最小化$|C|$;

(2) $\forall G\in C,\Sigma_{w\in G}Min\_D(w)\leq\min\{T_P(w)|w\in G\}$;

(3) C 是 W 的一个完全划分. (4)

为了证明问题(4)是 NP 难的,我们考虑如下与问题(4)等价的判定问题的 NP 完全性.

输入:集合 W,对于$\forall w\in W,Min_D(w),T_P(w),K$.

输出:是否存在 W 的一个大小为 K 的完全划分 C,使得$\forall G\in C$,

$$\Sigma_{w\in G}\ Min\_D(w)\le\min\{T_P(w)|w\in G\}. \tag{5}$$

定理 2. 问题(5)是 NP 完全问题.

证明:容易看到,问题(5)是 NP 问题,因为用非确定图灵机可以在多项式时间内穷举 W 的所有子集族,并判断每个子集族是否为 W 的完全划分,同时检查条件$\forall G\in C,\Sigma_{w\in G}\ Min_D(w)\le\min\{T_P(w)|w\in G\}$是否成立.下面,我们将多处理器调度问题[23]归约到问题(5).给定多处理器调度问题的任意实例 $M(A,l,D,m)$,其中 A 是任务集合,A 中任意任务 a 的长度 $l(a)$属于整数集合,所有任务的截止时间 D 属于整数集合,m 是处理器个数.令 $W=A$,W 中任意滑动窗口的 $Min\_D(w)=l(w),T_P(w)=D,K=m$,可以得到问题(5)的一个实例 $P(W,Min\_D,T_P,K)$.显然,上述转换过程可在多项式时间内完成.由多处理器调度问题和问题(5)的定义不难看出,多处理器调度问题的实例 $M(A,l,D,m)$有解当且仅当问题(5)的实例 $P(W,Min_D,T_P,K)$有解.由多处理器调度问题是 NP 完全问题可知,问题(5)是 NP 完全问题. □

显然,如果问题(3)可由确定的图灵机在多项式时间内求解,则问题(4)作为问题(3)的特例也可由确定的图灵机在多项式时间内求解,进而与问题(4)等价的问题(5)也可由确定的图灵机在多项式时间内求解.由于问题(5)是 NP 完全的,故问题(3)是 NP-Hard 问题.

4.3.2 动态规划算法

本节给出能够产生最优解的动态规划算法 C-Support-DP.此算法的思想是先将滑动窗口按照自然数进行编号,即对于$\forall w_i\in W$,令 $f(w_i)=i$,其中 $1\le i\le d$.然后,算法利用如下定义的集合函数 F 将 W 的任意一个子集一一映射为 1 与 2^d 之间的一个整数:$\forall A\subseteq W,F(A)=\Sigma_{w_i\in A}2^{f(w_i)}$.如果 $F(A)=n$,我们记 $A=F^{-1}(n)$.于是从 1 到 2^d 之间的任何一个整数唯一地对应了一个滑动窗口集合.基于以上表示方法,采用动态规划思想,我们可以递归地定义问题(3)的最优解代价.

设变量 $Is_Group[n]$标识滑动窗口集合 $F^{-1}(n)$是否为一个串行调整组,变量 $Cost[n]$表示集合 $F^{-1}(n)$的最优代价,变量 $split_point[n]$表示以滑动窗口集合 $F^{-1}(n)$为输入时问题(3)的优化解.于是,问题(3)优化解的代价方程为

(1) 若$|F^{-1}(n)|=1$ 且 $F^{-1}(n)=\{w\}$,则 $Is\_Group[n]$=true 且 $Cost[n]=Exch(w)$;

(2) 若 $Is\_Group[n]$=true,则 $Cost[n]=\max\{Exch(w)|w\in F^{-1}(n)\}$;

(3) 若 $Is\_Group[n]$=false,则 $Cost[n]=\min\{Cost[k]+Cost[n\ \text{XOR}\ k]|1\le k\le n$ 且 $F^{-1}(k)\subseteq F^{-1}(n)\}$.

下面给出求解问题(3)的动态规划算法 C-Support-DP.

输入:集合 W.

输出:集合 C,且 $C\subseteq 2^W$.

```
(1)  对滑动窗口按照自然数进行编号,即∀w_i∈W,令 f(w_i)=i,其中 1≤i≤d ;
(2)  for (i=0; i<d; i++)
(3)      n=2^i;   /*形成一个仅包含滑动窗口 w_i 的串行调整组 F^-1(n) */
(4)      Is_Group[n]=true, Cost[n]=Exch(i);   /*标识 F^-1(n)为一个串行调整组,并保存其共享内存量*/
(5)      for (j=1; j<n; j++)
(6)          m=n OR j;   /*将集合{w_i}与滑动窗口集合 F^-1(j)合并,形成一个新的滑动窗口集合 F^-1(m)*/
(7)          if (滑动窗口集合 F^-1(m)是一个串行调整组)
(8)              Is_Group[m]=true,Cost[m]=max{Cost[j], Exch(i)};   /*  解释同(4)*/
(9)          else
(10)             Is_Group[m]=false ;   /*标识 F^-1(m)不是一个串行调整组*/
(11)             Cost[m]= Cost[j]+Exch(i),split_point[m]=j;   /*记录划分 F^-1(m)的最优点*/
(12)             for (k=1;k<j;k++)
(13)                 tempcost=Cost[k]+Cost[m XOR k];   /*因为 F^-1(k)∪F^-1(m XOR k)=F^-1(m)*/
(14)                 if (j OR k = =j && Cost[m]>tempcost)   /*需要满足 F^-1(k)⊆F^-1(j)条件*/
(15)                     Cost[m]=tempcost,split_point[m]=k ;   /*解释同(11)*/
```

上述算法中,循环语句(2)执行 d 次,循环语句(5)执行 $n-1=2^i-1$ 次,循环语句(12)执行 $j-1$ 次.算法的时间复杂性为 $O(\Sigma_{i=0}^{d-1}\Sigma_{j=1}^{2^i-1}(j-1))=O\left(\frac{1}{2}\Sigma_{i=0}^{d-1}(2^{2i}-3*2^i+2)\right)=O\left(\frac{1}{6}4^d-\frac{3}{2}2^d+d+\frac{4}{3}\right)=O(4^d)$.算法中用到的数组长度均为 2^d,因此算法的空间复杂性为 $O(2^d)$.

输出串行调整组的算法描述如下.输出串行调整组算法的时间复杂性为 $O(d)$.

```
Get-Serial-Adjusting-Group (int n)
(1)  if (Is_Group[n]= =true)
(2)        输出串行调整组 F^{-1}(n);
(3)  else
(4)        Get-Serial-Adjusting-Group (split_point[n]) ;
(5)        Get-Serial-Adjusting-Group (n XOR split_point[n]);
```

4.3.3 近似调整算法

C-Support-DP 算法的复杂性很高,当 d 较大时,无法使用.然而,由于问题(3)是 NP-Hard 问题,我们无法给出具有多项式复杂性的准确优化算法.为此,本节给出一个具有多项式复杂性的近似算法 C-Support-AP. C-Support-AP 的基本思想是尽可能使交换容量大小相近的滑动窗口形成一个串行调整组,达到降低共享内存量总和的目的,同时也降低了所有滑动窗口占用的可用内存空间总量.算法 C-Support-AP 描述如下:

输入:集合 W.

输出:目标集合 $C\subseteq 2^W$.

```
(1)  按照交换容量大小降序排列所有滑动窗口,形成队列 w_1,...,w_i,...,w_d;   /* d 为滑动窗口个数*/
(2)  k=1; n=0;        /* k 表示滑动窗口的下标,n 标识已经产生的串行调整组个数*/
(3)  while (k≤d)
(4)        l=1;        /* l 表示串行调整组的下标*/
(5)        while (l ≤ n)
(6)              if (滑动窗口集合{w_k}∪Z_l 是一个串行调整组)   Z_l ={w_k}∪Z_l;   Break;
(7)              else l ++;
(8)        if (l > n) n++; Z_l ={w_k};   /*  如果滑动窗口 w_k 没有加入到已经产生的串行调整组*/
(9)        k++;
(10) 输出集合 C={Z_i|1≤i≤n}.
```

上述算法中,语句(1)的复杂性为 $O(d\log d)$,语句(3)和语句(5)形成两层循环,每层循环最多需要执行 d 次,因此,算法的时间复杂性为 $O(d^2)$.

请注意,当 $\Sigma_{w\in W}\ B(w)<0$ 时,如果优化算法或者近似算法产生的串行调整组集合为 C,而且 $\Sigma_{G\in C}\ Share(G)+\Sigma_{w\in W}\ Static(w)>M$,则滑动窗口规模的调整问题无解,即支持所有连续查询的可用内存空间的下界为 $\Sigma_{G\in C}\ Share(G)+\Sigma_{w\in W}\ Static(w)$.

5 调整滑动窗口规模的时机选择

本节讨论滑动窗口规模调整的时机选择,即触发调整滑动窗口规模的条件.滑动窗口规模调整的频率不宜过高,否则会降低系统的性能.滑动窗口的调整只需在特定的事件发生时进行.这些事件包括一个连续查询的开始或结束,数据流的流速发生变化等.具体的调整时机选择方法描述如下:

(1) 当一个连续查询开始或者结束时,系统分析该连续查询的特征,并根据当前时刻系统的可用内存空间分配状况,检查表 4 中的触发条件是否形成.若某一个触发条件成立,则调用相应的算法进行滑动窗口规模的调整.

(2) 当数据流的流速变化超出一定的阈值 δ 时,系统需要调用相应的算法进行滑动窗口规模的调整工作,其中阈值 δ 是一个经验值.

Table 4 Trigger conditions of adjusting window-width

表 4 调整滑动窗口规模的触发条件

State of q	State of system		
	$\Sigma_{w\in W}A(w)\geq 0$	$\Sigma_{w\in W}A(w)<0\leq\Sigma_{w\in W}B(w)$	$\Sigma_{w\in W}B(w)<0$
When q starts	$Q_t(q)>Max\_T(w)$	$Q_t(q)\geq Min\_T(w)$	$Q_{t\text{-}e}(q)\geq Min\_T(w)-Min\_D(w)$
When q ends	---	$Q_t(q)\geq Min\_T(w)$	$Q_{t\text{-}e}(q)\geq Min\_T(w)-Min\_D(w)$

6 实验结果

我们在具有 P4 2.4GHz CPU、256MB 主存、40GB 硬盘的微型计算机环境下进行了模拟实验.实验中的滑动窗口状态数据和连续查询的特征信息均随机生成.

我们首先来考察 B 级调整算法对查询误差的影响.实验中,我们随机地产生了 300 个和 600 个连续查询,用于考察共享内存量总和在实施滑动窗口规模调整之前和之后(应用 C-Support-DP 算法进行调整)的变化情况.实验结果如图 2、图 3 所示.从这两个图中可以看出,调整前后的滑动窗口累计误差总和变化比较明显,表明滑动窗口规模调整机制的使用在很大程度上减少了连续查询的误差总和.

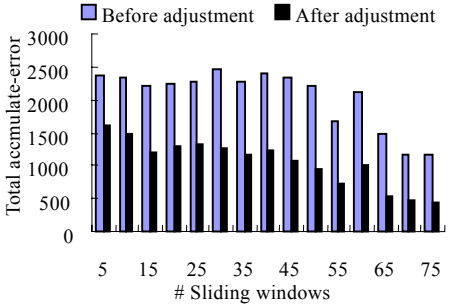

Fig.2 Total accumulate-error over 300 queries

图 2 处理 300 个查询时累计误差总和的变化情况

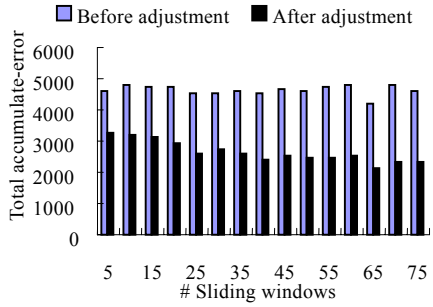

Fig.3 Total accumulate-error over 600 queries

图 3 处理 600 个查询时累计误差总和的变化情况

下面考察 C 级调整算法对内存利用率的影响.实验中,我们随机地产生了 300 个和 600 个连续查询,考察随着滑动窗口个数的变化,实施滑动窗口规模调整前后(应用 C-Support-DP 算法)的共享内存量总和的变化情况,实验结果如图 4、图 5 所示.使用调整机制前,不同的滑动窗口之间可能会在某时刻同时申请内存资源,造成滑动窗口对内存资源的需求增大,而使用调整机制后,同一个串行调整组内的多个滑动窗口之间以串行方式申请内存资源,因此调整机制的使用提高了内存空间的利用率.从这两个图中可以看出,使用调整机制前、后的共享内存量总和变化比较明显,而且随着滑动窗口个数的增多,变化逐渐增大,即调整机制节省的内存空间逐渐增大.

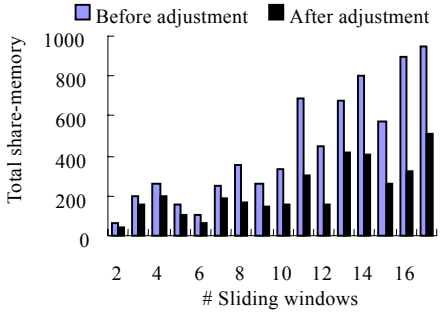

Fig.4 Total share-memory over 300 queries

图 4 处理 300 个查询时共享内存量总和的变化情况

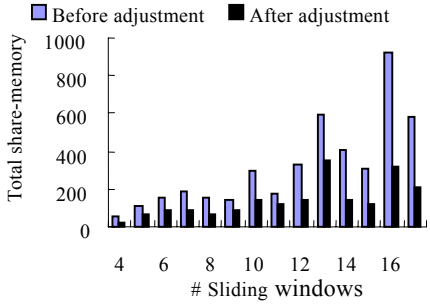

Fig.5 Total share-memory over 600 queries

图 5 处理 600 个查询时共享内存量总和的变化情况

下面比较算法C-Support-DP和算法C-Support-AP内存利用率的差异.实验中,我们随机产生了300个和600个连续查询,考察在滑动窗口个数不同的数据流系统中,算法 C-Support-DP 与算法 C-Support-AP 使用内存空间大小的绝对和相对差异.由于 C-Support-DP 最优化算法的时间、空间复杂性高,我们只能得到小规模问题的最优解,因此实验工作中滑动窗口的个数限制在 17 个以下.实验结果如图 6、图 7 所示.图中的相对差异定义为 $(SUM_{AP}-SUM_{DP})/SUM_{DP}$,其中 SUM_{AP} 是算法 C-Support-AP 消耗的共享内存量总和,SUM_{DP} 是算法 C-Support-DP 消耗的共享内存量总和.从这两个图中可以看出,不论在何种情况下,算法 C-Support-AP 与算法 C-Support-DP 之间的差异不是很大,近似算法的相对差异没有超过 20%.这说明求解大规模问题时,近似算法也可以取得很好的性能.

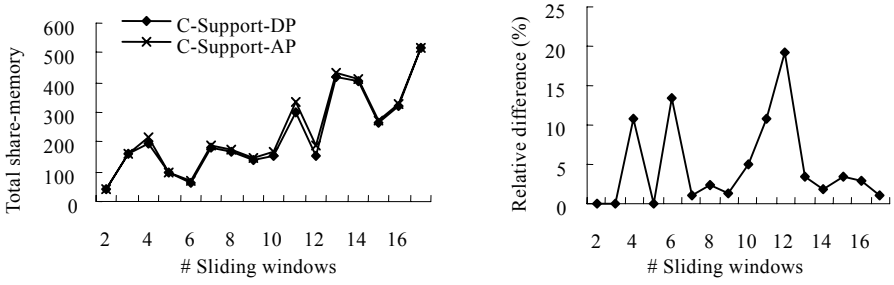

Fig.6 C-Support-DP vs. C-Support-AP over 300 continuous queries

图 6 处理 300 个连续查询时算法 C-Support-DP 和 C-Support-AP 的性能差异对比

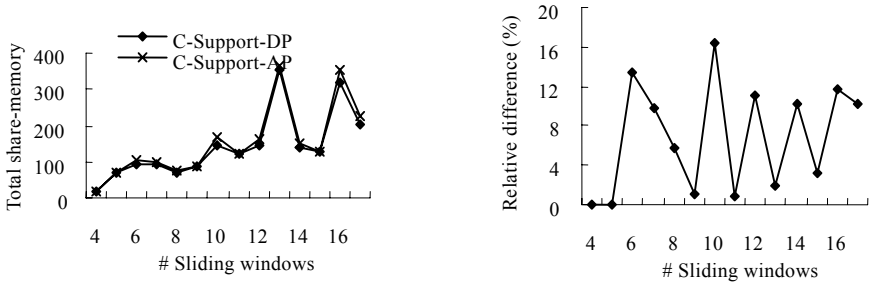

Fig.7 C-Support-DP vs. C-Support-AP over 600 continuous queries

图 7 处理 600 个连续查询时算法 C-Support-DP 和 C-Support-AP 的性能差异对比

7 结 论

本文提出了 3 类滑动窗口规模动态调整算法.理论分析与实验结果表明,本文提出的算法能够在有限的内存空间条件下,根据数据流的流速与连续查询的变化情况,有效地动态调整滑动窗口的规模,极大地提高了数据流上连续查询处理的质量.

References:

[1] Araru A, Babu S, Widom J. An abstract semantics and concrete language for continuous queries over streams and relations. Technical Report, Stanford University Database Group. 2002. http://dbpubs.stanford.edu/pub/2002-57

[2] Guha S, Koudas N. Approximating a data stream for querying and estimation: Algorithms and performance evaluation. In: Stefano C, Christoph F, Pat S, eds. Proc. of the 18th Int'l Conf. on Data Engineering. San Jose: IEEE Computer Society, 2002. 567~576.

[3] Golab L, Ozsu MT. Processing sliding window multi-joins in continuous queries over data streams. In: Freytag JC, Lockemann PC, Abiteboul S, eds. Proc. of the 29th Int'l Conf. on Very Large Data Bases. Berlin: Morgan Kaufmann Publishers, 2003. 500~511.

[4] Viglas SD, Naughton JF, Burger J. Maximizing the output rate of multi-way join queries over streaming information sources. In: Freytag JC, Lockemann PC, Abiteboul S, eds. Proc. of the 29th Int'l Conf. on Very Large Data Bases. Berlin: Morgan Kaufmann Publishers, 2003. 285~296.

[5] Hammad MA, Franklin MJ, Aref WG, Elmagarmid AK. Scheduling for shared window joins over data streams. In: Freytag JC, Lockemann PC, Abiteboul S, eds. Proc. of the 29th Int'l Conf. on Very Large Data Bases. Berlin: Morgan Kaufmann Publishers, 2003. 297~308.

[6] Babcock AK, Babu S, Datar M. Model and issues in data stream systems. In: Popa L, eds. Proc. of the 21st ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems. Madison: ACM, 2002. 1~16.

[7] Golab L, Ozsu MT. Issues in data stream management. SIGMOD Record, 2003,32(2):5~14.

[8] Motwani R, Widom J, Arasu A. Query processing, approximation, and resource management in a data stream management system. In: Proc. of the 1st Biennial Conf. on Innovative Data Syst. Res (CIDR). 2003. http://newdbpubs.stanford.edu/pub/2002-41

[9] Madden S, Franklin MJ. Fjording the stream: An architecture for queries over streaming sensor data. In: Proc. of the 18th Int'l Conf. on Data Engineering. San Jose: IEEE Computer Society, 2002. 555~566.

[10] Chandraskearan S, Franklin MJ. Streaming queries over streaming data. In: Bernstein PA, Loannidis YE, Ramakrishnan R, eds. Proc. of the 28th Int'l Conf. on Very Large Data Bases. Hong Kong SAR: Morgan Kaufmann Publishers, 2002. 203~214.

[11] Carney D, Cetintemel U, Cherniack M. Monitoring streams: A new class of data management applications. In: Bernstein PA, Loannidis YE, Ramakrishnan R, eds. Proc. of the 28th Int'l Conf. on Very Large Data Bases. Hong Kong SAR: Morgan Kaufmann Publishers, 2002. 215~226.

[12] Babcock B, Babu S, Datar M, Motwani R. Chain: Operator scheduling for memory minimization in data stream systems. In: Halevy AY, Ives ZG, Doan A, eds. Proc. of the 2003 ACM SIGMOD Int'l Conf. on Management of Data. San Diego: ACM, 2003. 253~264.

[13] Olston C, Jiang J, Widom J. Adaptive filters for continuous queries over distributed data streams. In: Halevy AY, Ives ZG, Doan A, eds. Proc. of the 2003 ACM SIGMOD Int'l Conf. on Management of Data. San Diego: ACM, 2003. 563~574.

[14] Madden S, Shah M, Hellerstein JM, Raman V. Continuously adaptive continuous queries over streams. In: Franklin MJ, Moon B, Ailamaki A, eds. Proc. of the 2002 ACM SIGMOD Int'l Conf. on Management of Data. Madison: ACM, 2002. 49~60.

[15] Tatbul N, Cetintemel U, Zdonik S. Load shedding in a data stream manager. In: Freytag JC, Lockemann PC, Abiteboul S, eds. Proc. of the 29th Int'l Conf. on Very Large Data Bases, Berlin: Morgan Kaufmann Publishers, 2003. 309~320.

[16] Carney D, Cetintemel U, Rasin A, Zdonik S, Cherniack M, Stonebraker M. Operator scheduling in a data stream manager. In: Freytag JC, Lockemann PC, Abiteboul S, eds. Proc. of the 29th Int'l Conf. on Very Large Data Bases. Berlin: Morgan Kaufmann Publishers, 2003. 838~849.

[17] Cherniack M, Balakrishnan H, Balazinska M. Scalable distributed stream processing. In: Proc. of the 1st Biennial Conf. on Innovative Data Syst. Res (CIDR). 2003. http://nms.lcs.mit.edu/papers/medusa-cidr03.html

[18] Arasu A, Babcock B, Babu S. Characterizing memory requirements for queries over continuous data streams. In: Popa L, eds. Proc. of the 21st ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems. Madison: ACM, 2002. 221~232.

[19] Kang J, Naughton JF, Viglas SD. Evaluating window joins over unbounded streams. In: Umeshwar D, Krithi R, Vijayaraman TM, eds. Proc. of the 19th Int'l Conf. on Data Engineering. Bangalore: IEEE Computer Society, 2003. 341~352.

[20] Haas PJ. Large-Sample and deterministic confidence intervals for online aggregation. In: Proc. of the 9th Int'l Conf. on Scientific and Statistical Database Management (SSDBM, 1997). 1997. 51~63.

[21] Hellerstein JM, Hass PJ, Wang HJ. Online aggregation. In: Peckham J, ed. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. Tucson: ACM Press, 1997. 171~182.

[22] Babcock B, Datar M, Motwani R. Load shedding for aggregation queries over data streams. In: Rundensteiner E. Proc. of the 20th Int'l Conf. on Data Engineering. Boston: IEEE Computer Society, 2004. 350~361.

[23] Garey MR, Johnson DS. Computers and Intractability: A Guide to the Theory of NP-Completeness. Bell Laboratories, 1978.