文章编号:1001-2486(2007)04-0047-05

一种时空数据流中范围聚集查询共享策略*

左怀玉,靳肖闪,景 宁,钟志农
(国防科技大学 电子科学与工程学院,湖南 长沙 410073)

摘 要:根据查询谓词的相似性来实现计算共享是数据流查询优化的主要策略之一,主要针对时空数据流中基于滑动窗口的范围聚集问题进行查询优化,采用时间片段和空间片段分别描述滑动窗口之间和空间范围谓词之间的重叠。提出基于时空分片技术将时空数据流元组分成互不重叠的时空片段。范围聚集查询首先基于时空片段生成中间结果,然后根据查询条件与时空片段的映射关系生成查询结果。实验表明,提出的时空分片策略能大大提高时空数据流中范围聚集查询的性能。

关键词:时空数据流;范围聚集查询;查询优化
中图分类号:TP391 **文献标识码**:A

Sharing for Multiple Range Aggregation Queries over Spatio-temporal Streams

ZUO Huai-yu, JIN Xiao-shan, JING Ning, ZHONG Zhi-nong
(College of Electronic Science and Engineering, National Univ. of Defense Technology, Changsha 410073, China)

Abstract: By exploiting query predicate similarities in the data streams, queries optimization is a mainly strategy to share computation. In this paper, based on the sliding windows, an approach is presented for the multiple range aggregation queries over spatio-temporal streams. Time fragments were employed to describe the overlaps between sliding windows and spatial fragments for the overlaps between spatial range predications. Then a technique was proposed to partition the streams into different spatio-temporal fragments without any overlaps. The final query results could be captured from the temporary query results over the spatio-temporal fragments. Experimental result shows that our approach can substantially increase performance of the multiple range aggregation queries over spatio-temporal streams.

Key words: spatio-temporal data stream; multiple range aggregation queries; query optimization

针对位置感知环境中移动对象产生的位置属性数据,其查询处理目前主要有两类方式,其一是假定移动对象产生的数据可以在二级存储设备上物化,查询处理采用各种扩展存储索引来提高性能,如TPR-tree及其变种[1]等时空数据索引方法;其二是将移动对象产生的数据看成是时空数据流[2],利用数据流技术来实时处理。目前后者正受到越来越多的关注,典型的时空数据流管理系统如PLACE[3]、CAPE[4]等已能有效处理环境感知条件下移动对象产生的位置数据。

空间范围聚集查询是针对时空数据流中元组空间属性的常用查询方式之一,它从时空数据流中获取满足空间范围查询谓词的对象元组集合,并返回一个关于集合中移动对象非空间属性的聚集信息。本文针对时空数据流中范围聚集查询提出了一种实时共享计算技术,它通过将时间和空间划分为不同层次的共享区域,并在共享区域的基础上构建查询中间结果,大量并发的范围聚集查询则通过这些共享区域的中间结果生成最终查询结果。

1 相关工作

传统关系数据库管理系统中的查询多是一次查询,其资源共享策略通常在不同查询中寻求公共查

* 收稿日期:2006-12-25
基金项目:国家自然科学基金资助项目(60472031);国家863高技术计划资助项目(2006AA701312);高等学校博士学科点专项基金资助项目(20059998012)
作者简介:左怀玉(1978—),男,博士生。

询条件来实现,这种技术常见于多路查询优化[5]。在持续查询环境下,NiagaraCQ[6]系统采用基于查询表达结构进行分组共享,首先寻找不同查询表达的公共内容,形成分组描述,其次将不同的分组描述以XML文件形式存储到分组表中,共享策略则基于分组表中的公共或相似查询条件实现,其实质是基于公共查询条件来实现资源共享。

基于查询算子的相似性来实现资源共享是数据流持续查询优化的另一个主要策略。大多数的研究工作都集中于过滤查询,如文献[6]采用谓词索引在过滤查询中实现资源共享。基于XML的过滤查询[7]和近似连接处理中采用共享sketches[8]等都属于这类技术。Arasu等[9]则提出在基于滑动窗口中的聚集查询中采用分而治之的思想将聚集查询结果分不同层次时段预先计算出来,时段长度为$(i\cdot 2^l+1,(i+1)\cdot 2^l)$,其中$l$表示层次。大量并发的聚集查询则共享这些不同层次时段上的计算结果。Li[13]采用时间分片思想将滑动窗口划分为众多称之为panes的小窗口,聚集查询共享这些时间片段形成最终结果。其思想和Arasu[9]相似。

2 基于时空分片的聚集共享

2.1 问题描述

本文考虑的聚集函数g包含distributive和algebraic两类[12]。假定X_1,X_2表示聚集函数定义域,上述聚集函数具有性质$g(X_1\cup X_2)=f(l(X_1)\cup l(X_2))$,其中$l$为部分聚集函数,$f$为整体聚集函数。这个性质是构成范围聚集查询共享的基础。由于时空数据流的无界性,聚集查询通常使用滑动窗口来消除谓词的阻塞影响,滑动窗口使用RANGE参数表明其查询时间段,使用SLIDE参数表明其查询结果更新间隔。我们用$w(r,s)$表示RANGE为r和SLIDE为s的滑动窗口,根据r和s之间的大小关系,滑动窗口可以分为三类。当$r>s$时,滑动窗口的聚集查询最复杂,此时称为重叠滑动窗口。

定义1 令$W=S[T_L,T_R]_T$表示时空数据流S中滑动窗口,其元组形式为$p_i=<loc_i,s_i,\tau_i>$,A为S模式上的属性,其值域为D。对于聚集函数$g\in Agg$,有$g_A:D\to D_{Agg}$,D_{Agg}表示聚集函数g的值域。则查询$g_A(q,W)=g_A(Sel(P)_W)=g_A(\{p_i\mid \exists p_i\in W,p_i.loc\in q_R\})$表示属性$A$上的一个范围聚集查询。

2.2 基于时间分片的聚集共享

假定查询Q_1和Q_2表示两个不同的滑动窗口W_1和W_2上的范围聚集查询,根据定义1,$Q_1=g_A(\{p_i\mid \exists p_i\in W_1,p_i.loc\in q_{R_1}\})$,$Q_2=g_A(\{p_i\mid \exists p_i\in W_2,p_i.loc\in q_{R_2}\})$,其中滑动窗口的RANGE和SLIDE参数分别是$r_i,s_i$。本节提出一种基于时间片段的思想来实现$Q_1$和$Q_2$之间的计算共享。

(1)单个聚集查询的时间片段聚集共享

我们的思想是将输入时空数据流分割成互不重叠的部分,称之为时间片段。时间片段上的聚集结果称之为部分聚集函数(partial aggregation),这些部分聚集函数可以通过整体聚集函数(final aggregation)形成查询结果。

基于滑动窗口的查询具有周期性且其频率受SLIDE支配,因此我们的目标是将RANGE(r_i)表达为SLIDE(s_i)函数。不失一般性,令$r_i=\alpha\cdot s_i+\beta$,则$\beta=r_i-\alpha\cdot s_i$,即$\beta=r_i \bmod s_i$。令$s_i=\beta+\upsilon$,则RANGE和SLIDE的最终关系可以描述为:

$$\begin{cases} r_i=(\alpha+1)\beta+\alpha\upsilon \\ s_i=\beta+\upsilon \end{cases}$$

定理1[12] 滑动窗口$W(r,s)$上的聚集可以通过其时间片段$W(\beta,\upsilon)$上的聚集结果获取。

假定滑动窗口$W(r,s)$上元组个数为N,时间片段β和υ上的元组个数分别是$\sum_{0\leqslant i\leqslant\alpha+1}N_{\beta_i}$和$\sum_{0\leqslant i\leqslant\alpha}N_{\upsilon_i}$,如果不采用时间分片来共享计算,则代价为$\alpha\cdot(\sum_{0\leqslant i\leqslant\alpha+1}N_{\beta_i}+\sum_{0\leqslant i\leqslant\alpha}N_{\upsilon_i})$次聚集计算;采用时间片段只需要计算$N+2\alpha+1$次聚集计算。其中$N$是部分聚集函数计算次数,$2\alpha+1$是整体聚集函数次数。

(2)多个聚集查询的时间片段聚集共享

当 Q_1 和 Q_2 同时注册为查询计划后,仅仅依靠单个聚集查询的时间片段来共享计算将忽略掉它们之间的相关性。如图 1 所示,图中重叠不仅发生在每个滑动窗口自身,而且还发生在滑动窗口与滑动窗口之间。对于图中的重叠区域需要重新划分时间片段来消除多余的重复计算。

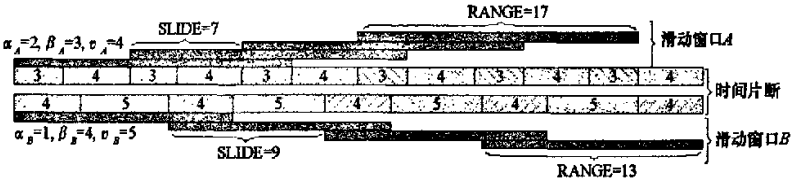

图 1 滑动窗口之间的重叠

Fig.1 Overlaps between sliding windows

从图 1 中可以看出,依照每个滑动窗口已划分时间片段的边界可以重新对时间进行划分,但这种划分需要对每个滑动窗口所属时间片段的边界进行排序。假定图 1 中滑动窗口 A 和滑动窗口 B 起始点相同,均为 0。设 M 为 SLIDE_A 和 SLIDE_B 的最小公倍数,则多个聚集查询的时间片段划分只需要在[0, M]内进行,可以看出,这种时间片段的划分也具有周期性。设计了如下的算法来将滑动窗口的时间划分为互不重叠的时间片段:

Algorithm 1: Time Fragments Manager
Input: Sliding Windows $\{SW_i\}$
Output: Time Fragments' Edges
Proc MultiWindowTimeFragments($\{SW_i\}$
  //计算最小公倍数
  M = LowestCommonMultiple($\{r_i\}$, $\{s_i\}$)
  For each SW in $\{SW_i\}$
    //计算单个滑动窗口的时间片段
    T_i = SingleWindowTimeFragments(SW, M)
  End for
  //对所有滑动窗口的时间片段排序
  T = Sorted($\{T_i\}$)
  Return T

Proc Single WindowTimeFragments (SW, m)
  a = r mod s
  b = s − a
  t[m]{}
  For i = 0 to m
    $t_i = \sum_{n=0}^{m}\left(\lceil\frac{n+1}{2}\rceil a + \lceil\frac{n-1}{2}\rceil b\right)$
  End for
  Return t[m]

2.3 基于空间分片的聚集共享

范围聚集查询中的空间范围谓词之间也可以根据空间区域实现计算共享。令 $Q=\{Q_i\}(1\leqslant i\leqslant N)$,所有空间范围谓词表示的区域集合为$\{q_{R_1},\cdots,q_{R_N}\}$,这些区域将时空数据流划分为互不重叠的空间片段,记为$\{F_0,\cdots,F_k\}$。假定元组个数为 T,则有 $T=F_0\cup\cdots\cup F_k$。

为实时实现空间分片技术,我们对注册查询集合 Q 中所有空间范围矩形区域进行水平和垂直排序。当时空数据流中元组 t 到达时,首先在水平方向上进行二分查找确定 $t.loc$ 的所属区域,然后在垂直方向上进行二分查找确定 $t.loc$ 的所属区域。设 t 最终落入的区域是 F_i,则根据 F_i 和 Q 中空间范围谓词之间的关系可以确定 t 对查询集合 Q 的作用。

定义 2 设空间分片结果为 $F=\{F_0,\cdots,F_k\}$,令 $b=\{b_1,\cdots,b_N\}(b_j=\{0,1\})$,当且仅当时空数据流中的元组 t 在查询过程中满足 $Q_j\in Q(F_i)$时,$b_j=1(1\leqslant j\leqslant N)$,称 b 为元组 t 的位图因子。

对于每一个元组 t,附加 N 比特位图因子描述元组 t 和所属范围聚集查询之间的关系,记作(t,b),并称之为空间分片元组。容易得到$\{b\}$的值域个数和空间分片个数一致,都是 k。位图因子实质上是记录了元组 t 和查询集合 Q 之间的对应关系。时空分片算法如下:

```
Algorithm 2: Spatial Fragments Manager
Input: Tuple t, Range Rectangle Set {R_1, ..., R_N}
Output: Spatial Fragments Tuple (t, b)
Proc SpatioFragments (t, {R_j})
  {X_i}{Y_i}   //矩形水平和垂直坐标
  {b_i^X}{b_i^Y}   //水平和垂直位图因子
  {X_i} = Sorted({R_j.Left, R_j.Right});   //所有矩形水平坐标排序
  {Y_i} = Sorted({R_j.Top, R_j.Bottom});   //所有矩形垂直坐标排序
  For each b_i in {b_i^X}
    For each byte b in b_i
        If X_i >= R_j.Left and X_{i+1} < R_j.Right then
          b = 1
        else
          b = 0
        End if
    End for
  End for
  m = GetPosition(X_i, t.loc.x)   //判断元组水平坐标位置
  n = GetPosition(Y_i, t.loc.y)   //判断元组垂直坐标位置
Return(t, b_m^X and b_m^Y)
```

在输出端设置 Hash 桶来计算带有不同位图因子 b 的空间分片元组部分聚集函数，对于每一个部分聚集函数结果记作(l_i, b_i)。整体聚集函数根据 b 的第j 位来计算查询 Q_i 的结果。

2.4 基于时空分片的聚集共享

将时间分片和时空分片思想结合起来，构建了基于时空分片的空间范围聚集共享，图 2 描述了时空分片原理。

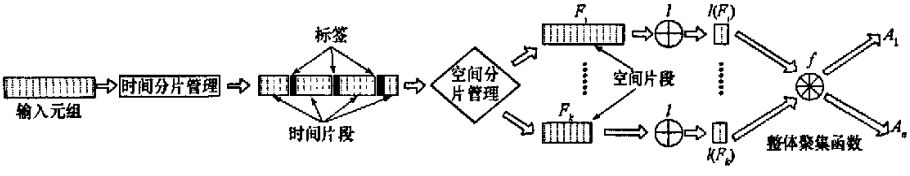

图 2 空间范围聚集查询的时空分片

Fig.2 Spatio-temporal fragments for multiple range aggregation queries

对于时空数据流中的输入元组，在时间分片管理中根据已注册空间范围聚集查询集合 Q 调用算法 1 对每个滑动窗口进行时间分片，时间分片结果采用嵌入标签的方式表达分片信息，标签模式和元组保持一致，其时间戳保证不破坏原数据流中的时间序列性。空间分片管理调用算法 2 对每个查询条件中的矩形区域进行空间分片，并在空间分片结果中嵌入位图因子描述元组和查询 Q_i 之间的关系。空间范围聚集查询计算在时间分片和空间分片基础上分别进行，输出端根据时间标签和位图因子组合成最终结果。

3 实验及性能分析

3.1 实验内容及设置

实验模拟数据采用预先随机生成点数据均匀分布在 10 000 × 10 000 空间域内，点数据个数为 1 000 000，将其保存到一台客户机上，客户机顺序读取这些数据并发送到查询执行主机上，时间序列统一在客户机上根据当前系统时间生成。空间范围聚集查询模式为：

SELECT count(s) FROM s [RANGE r min SLIDE s] WHERE s.*point* in Region R

查询中涉及到的 RANGE 等参数均在主机上采用预先随机生成，满足参数 $s = \lceil r/3 \rceil$。r 在[3,20]中

随机选取,矩形 R 的长度和宽度在[500,5000]中随机选取,矩形中心均匀散落在模拟数据的 $N*N$ ($5000<N<50\,000$)空间域 V 内,V 和背景区域中心重合;空间范围聚集查询个数共计 300 个。查询性能估计采用回答所有空间范围聚集查询的平均执行时间衡量。

3.2 实验结果与分析

实验 1 的目的是验证时空分片技术在查询个数增多的情况下的适应性。图 3 描述了数据流流速为 1.5×10^3/s,模拟数据区域参数 N 为 5000 的查询个数和执行时间的关系曲线。在查询个数较少的情况下,时空分片算法并没有明显优势,相反其性能比传统算法还要稍微逊色。随着查询个数的增加,时空分片算法的耗时代价明显优于传统算法。当查询个数为 300 时,时空分片算法耗时仅仅是传统算法的 45%左右。

实验 2 的目的是验证时空分片技术在流速增快情况下的适应性。图 4 比较了数据流流速变更条件下传统算法和时空分片算法的性能。图中查询个数固定为 256,模拟数据区域参数 N 固定为 5000。传统算法和时空分片算法和数据流流速均近似具有线性关系,但时空分片算法的梯度明显小于传统算法。此时时空分片的代价在总体代价中所占比例较小,所以对线性难有影响。

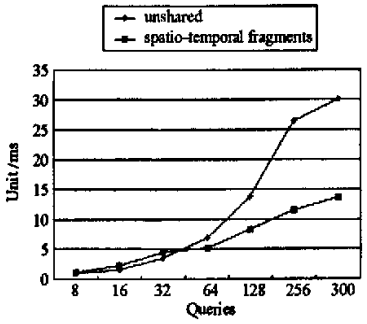

图 3 改变查询个数时的性能比较
Fig.3 Unshared/spatio-temporal fragments with different queries

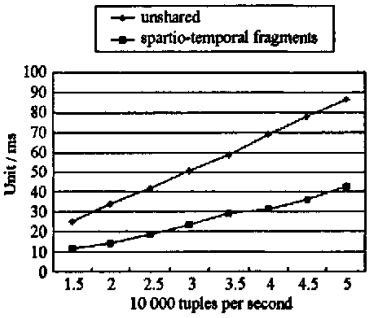

图 4 改变数据流流速时的性能比较
Fig.4 Unshared/spatio-temporal fragments with different data stream rates

4 总结

针对时空数据流中的范围聚集查询,提出了一种基于时空分片策略的共享计算方法。与现有数据流中基于时间分片的各种传统共享计算相比,我们增加了针对空间范围谓词的空间分片技术,最终形成的基于时空分片的共享计算技术能很好适应数据流中查询条件的注册和撤销。实验结果表明基于时空分片技术的查询共享计算性能高于传统算法,且对空间分片重叠覆盖率参数较为敏感,能有效改善系统性能。

参考文献:

[1] Saltenis S, Jensen C. S. Indexing of Moving Objects for Location-based Services[C]//ICDE'02, 2002.
[2] Hicham G E. Challenges in Spatio-temporal Stream Query Optimization[C]//MobiDE'06, 2006.
[3] Mokbel M F, Xiong X P, Hammand M A, et al. Continuous Query Processing of Spatio-temporal Data Streams in PLACE[C]//WSDM'04, 2004.
[4] Rundensteiner E A. CAPE: Continuous Query Engine with Heterogeneous-grained Adaptivity[C]//VLDB'04, 2004.
[5] Roy P, Seshadri S, Sudarshan S, et al. Efficient and Extensible Algorithms for Multi-query Optimization[C]//ACM SIGMOD, 2000.
[6] Chen J, DeWitt D J, Tian F, et al. NiagaraCQ: An Acalable Continuous Query System for Internet Databases[C]//ACM SIGMOD, 2000.
[7] Gupta A K, Suciu D. Stream Processing of XPath Queries with Predicates[C]//ACM SIGMOD, 2003.
[8] Dobra A, Garofalakis M, Gehrke J, et al. Sketch-based Multi-query Processing over Data Streams[C]//EDBT'04, 2004.
[9] Arasu A. Continuous Queries over Data Streams[D]. Ph. D Dissertation, 2006:132 - 163.
[10] Mokbel M F, Xiong X, Aref W G. SINA: Scalable Incremental Processing of Continuous Queries in Spatio-temporal Databases[C]//ACM SIGMOD, 2004.
[11] Xiong X, Mokbel M F, Aref W G, et al. Scalable Spatio-temporal Continuous Query Processing for Location-aware Services[C]//SSDBM'04, 2004.
[12] Gray J. Data Cube: A Relational Aggregation Operator Generalizing Group-by, Cross-tab and Sub-total[C]//ICDE'96, 1996.
[13] Li J. No Pane, No Gain: Efficient Evaluation of Sliding-window Aggregates over Data Streams[C]//ACM SIGMOD, 2005.