文章编号: 1001 - 9081(2008) S1 - 0169 - 03

一种动态维护分布式环境下 top-k集合的近似算法

田海生[1],陈立军[2],邱海艳[2],赵 静[2]
(1.中国科学院 研究生院,北京 100039; 2.北京大学 计算机系,北京 100871)
(thaisheng@126.com)

摘 要:在分布式数据流场景中,如何动态维护 top-k集合并尽可能地降低通信开销是非常重要的。通常的做法是:把大量的数据从分布式节点传送到中央节点,然后在中央节点计算 top-k集合。这样的通信开销非常大,在许多场合下是根本无法实现的。提出了一种高效地动态维护分布式环境下 top-k集合的近似算法 top-k′。在算法中对一个 top-k查询,通过动态维护 $k'(k_{max} \geq k' \geq k)$个最高积分的元组,可以从中选取积分最高的 k个元组返回。实验表明 top-k′显著降低了各节点与中央协调节点之间的通信代价。

关键词: top-k′; top-k; 分布式数据流
中图分类号: TP311 **文献标志码**: A

top-k′: Approximate algorithm of maintaining dynamic top-k sets in distributed environment

TIAN Hai-sheng[1], CHEN Li-jun[2], QIU Hai-yan[2], ZHAO Jing[2]
(1. *Graduate University of Chinese Academy of Sciences, Beijing* 100049, *China*;
2. *Department of Computer Science & Technology, Peking University, Beijing* 100871, *China*)

Abstract: It is very important to maintain top-k aggregate dynamically and reduce the cost of communications, in a typical distributed data stream scenario. In general, we need to continuously transmit data from many different branches to central coordinator, and then calculate the results of top-k. But it costs a great deal of communications, and it is unacceptable in many cases. In this article, according to other's results, a new high effective approximate algorithm top-k′ was proposed which was able to maintain a dynamic top-k aggregate in distributed data environment. According to our algorithm, by dynamically keeping $k'(k_{max} \geq k' \geq k)$ data elements which had the highest scores, the highest scores of the k data elements from those could be chosen as a result of top-k query. The tests indicate that our method dramatically reduces the cost of communications between branches and central coordinator.

Key words: top-k′; top-k; distributed data stream

0 引言

近年来,随着对数据流上查询处理的研究的深入,如何支持分布式数据流上 top-k查询[1]成为研究的热点和难点。top-k查询广泛使用于在线监测应用中,例如传感网络、证券分析、市场决策、电信欺诈监测、网络日志分析等。以分布式 Web服务器日志监测为例。1998年 FIFA世界杯网站自 1998年 4月 30日到 7月 26日接受了超过 10亿次访问[2],平均每分钟超过 11 000次。网站由分布在 4个不同地理位置的 30台服务器构成,用户访问通过 Cisco分布式控制器发送给不同的服务器处理。基于网站的访问日志,可以提出如下监测查询:

查询 1 目前在所有的服务器上最热门的网页是哪些?

查询 2 在某个地理区域的本地机群中,当前负载最小的是哪台服务器?

对于上面的两个查询,可以把所有的数据发送到一个中央节点,然后在中央节点上根据这些数据动态维护 top-k集合。但是这样做的通信开销非常大,在许多时效强的场合是无法接受的,而且发送所有数据也是没有必要的。在文献[1]中提出了一种通过少量的通信来动态维护分布式环境下 top-k集合的算法。在大部分情况下维护 top-k′$(k_{max} \geq k' \geq k)$的集合比维护 top-k的通信开销要小得多。基于此,本文提出了 top-k′算法,它通过动态维护 $k'(k_{max} \geq k' \geq k)$个最高积分元组的集合来支持 top-k查询。采用本算法,能够在提高查询效率的基础上提供近似的 top-k查询。

1 相关工作

top-k查询的研究已经取得了很大的成果[1,3-7]。在流上进行 top-k查询监视,首先要有一个初始计算的 top-k集合,然后在此基础上进行动态维护。文献[6-7]中提出了两种比较经典的计算初始 top-k集合的算法。文献[6]中提出了 TA算法,该算法的最大缺点就是通信次数不确定,而通常情况下通信次数会比较多。文献[7]提出了 TUPT算法,该算法分三个阶段完成 top-k的计算,与 TA算法相比通信次数较固定,算法较稳定。在对 top-k集合动态维护方面:文献[1]和[3]都是通过维护与 top-k相关的视图来达到回答 top-k查询的目的。文献[5]主要关注于如何找到与 top-k查询最相关的视图以及如何把视图信息用于 top-k查询;而文献[3]中更多地关注如何降低通信和时间开销来尽量快地回答用户的查询,它提出了一种通过维护 top-k视图来达到回答 top-k查询的目的,这个算法

收稿日期: 2007-08-02;**修回日期**: 2007-10-11。

作者简介:田海生(1973-),男,山东高密人,硕士,主要研究方向:数据库; 陈立军(1968-),男,教授,博士,主要研究方向:数据流处理、传感网络; 邱海艳,女,硕士,主要研究方向:数据库、信息系统; 赵静,女,硕士,主要研究方向:数据库、信息系统。

虽然增加了少量的空间开销,但是在通信和时间开销较大地降低。文献[4]提出了一种高效地动态维护数据流滑动窗口上的 top-k集合的算法,但是这个算法仅仅限于集中式环境下的数据流。文献[1]通过在每个节点上维护一系列约束在很大程度上降低了通信开销,达到了高效维护分布式环境下的 top-k集合的目的。该算法为初始的集合 T分配调节量并向每个监视节点发送 T和相应的调节量,各个监视节接收到这些信息后就持续地监视各自的约束条件,如果每个监视节点都满足约束条件,top-k集合不变,否则,重算 T和调节量,如此反复。本文在此基础上进行了改进,提高了效率。

2 相关背景知识

2.1 top-k查询

假设某分布式监视环境下有 $(m+1)$个节点,其中1个中央协调节点 N_0,m个远程监视节点 N_1, N_2, …, N_m。这些监视节点监视 n个逻辑对象 $U=\{O_1, O_2, \cdots, O_n\}$,每个逻辑对象对应一个聚合函数值 V_1, V_2, …, V_n。N_j不断产生变量序列 $S:\langle O_i, N_j, \Delta\rangle$,这表示监视节点 N_j监测到对象 O_i发生了大小为 Δ的变化,而这个变化只有 N_j可以看到,这样,N_j看到的 O_i的值就只是 $V_{ij}=\sum_{\langle O_i,N_j,\Delta\rangle\in S}\Delta$。在上面的基础上,top-$k$查询就是求 U的一个子集 T,使得 $|T|=k$,且有:

$$\begin{cases}\forall O_t\in T\\ \forall O_s\in U-T\\ V_t+\varepsilon\geq V_s\end{cases}$$

其中 $\varepsilon\geq 0$,是由用户指定的误差系数,当 $\varepsilon=0$时表示求精确解。

2.2 相关概念定义

对应每一个值 V_{ij}有一个调节量(adjustment factor) δ_{ij},实际上,调节量是对数据值在各个监视节点上的重新分配,经过这样的重新分配,系统可以保证各个监视节点上的 top-k集合 T和中央维护节点上的 top-k集合一致。监视节点 j上的产生变化后,如果调节量依然满足约束:

$$\begin{cases}\forall O_t\in T\\ \forall O_s\in U-T\\ V_t+\delta_{tj}>V_{sj}+\delta_{sj}\\ 1\leq j\leq m\end{cases}$$

则 top-k集合不变,否则,意味着 top-k集合发生了变化,应该重新计算 top-k集合并重新分配调节量。

基于以上概念,本文定义了两个边界:

$$G_j=\min_{O_t\in T}(V_{tj}+\delta_{tj})$$

$$H_j=\max_{O_s\in(U-T)}(V_{sj}+\delta_{sj})$$

其中 $1\leq j\leq m$。很显然,G_j代表在 j监视节点上,在 T中的各个对象的最小值,本文称之为最小 k值;而 H_j代表在 j监视节点上,不在 T中的各个对象的最大值,本文称之为最大非 k值。

这样,只有两种情况下会出现违反约束的变化:

1) $\langle O_s, N_j, \Delta\rangle, O_s\in(U-T), \Delta>0, V_{sj}+\delta_{sj}+\Delta>G_j$

2) $\langle O_t, N_j, \Delta\rangle, O_t\in T, \Delta<0, V_{tj}+\delta_{tj}+\Delta<H_j$

在第1种情况下,目前不在 top-k集合中的数据对象 O_s增大,有可能进入 top-k集合。如果经过计算,O_s进入 top-k集合,同时原有 top-k集合中值最小的数据对象退出 top-k集合,本文称这种 top-k集合的变化为 top-k集合的竞争,称 O_s为竞争者。

在第2种情况下,目前在 top-k集合中的数据对象 O_t减小,有可能退出 top-k集合。如果经过计算,O_t退出 top-k集合,同时原来不在 top-k集合中值最大的数据对象进入 top-k集合,本文称这种 top-k集合的变化为 top-k集合的补充,称 O_t为退出者。

3 自适应 top-k算法

3.1 算法提出的背景

文献[1]中提出来的维护算法在很大程度上降低了通信开销,但根据其进行的实验中,研究人员发现了一个比较有趣的现象:分布式 top-k查询的通信开销,对不同的 k会有差异,甚至是大小相差不大的 k_1和 k_2($|k_1-k_2|\ll k_1$),通信开销却相差很大。通过分析可以知道,这是由数据集的特点决定的。例如某个数据流在一段时间内排名第10和第11的两个数据对象 O_i和 O_j的值交替增长,这样它们的排名也经常交换;那么在这段时间内维护 Top 10的通信开销无疑会比维护 Top 11要大得多。

从这一发现出发,如果可以在 top-k算子的执行过程中自适应地选择维护通信开销较小的 top-k'($k'\geq k$)集合,就可以较大地提高算法效率。从应用的角度来说,采用这个维护算法之后,top-k查询得到的结果是前 k个值最大的数据对象,相当于在 k上引入了误差;但是只要误差能够控制在一定的范围,即 $k'\leq k_{\max}$,对于大部分应用来说还是可以接受的。

3.2 算法框架

图1介绍了本文进行分布式 top-k处理的系统框架。缓冲区负责中央维护节点和监视节点的通信。维护模块调整量的分配、监视节点的变化信息都通过缓冲区来发送和缓存。预处理模块负责生成初始的 top-k集合。维护模块是这篇文章主要关注的地方,当某个监视节点不满足其约束的时候,维护模块会动态地更新 top-k对象集合。

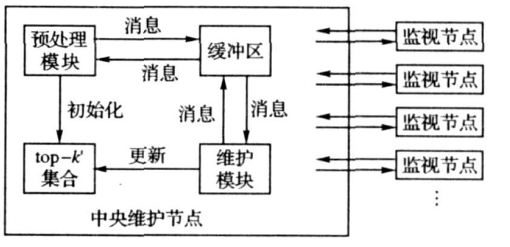

图1 系统框架

3.3 自适应 top-k算法

在 top-k集合的动态维护过程中,竞争和补充频繁交替进行,就会导致系统的通信开销非常大。如果本文把 k的值适当放大,使这些竞争和补充变成 top-k集合的内部数据对象的之间的排名变化,就能较大地减少 top-k集合的变化次数,从而减少通信开销。

top-k算法中前4步由中央协调节点 N_0完成,第5～7步由每个监视节点 N_j分别完成,第9步由 N_0和 N_j共同协作完成,这一步也是 top-k算法的核心。

算法1 top-k算法

Begin

1) 初始化 $k'=k$;
2) 使用任何可用的算法计算初始的 top-k集合 T;
3) 调用 $AFR[1]$,为初始的 T计算调整量 δ_{ij};
4) 向监视节点 N_j发送集合 T和 δ_{ij}
5) 各监视节点计算各自的 G_j和 H_j;
6) 各监视节点 N_j不停地监视各自的约束
7) if(每个对象 O_i都满足约束) then

8)　　top-k 集合不变;
9)　　else
10)　　调用算法 adjust调整集合 T、调整量 $\delta_{i,j}$、K 以及各个 G_j 和 H_j
11)　　goto 6)
End

3.4　adjust算法

在监视节点发现有不满足约束的对象出现的时候,系统会在第 9步中调用算法 adjust来进行重新调整,由于调整过程需要中央协调节点和分布式监视节点的共同参与,因此这是一个分布式的算法。在本文的设计中,adjust算法由三个阶段组成。

阶段 1　约束失效的节点 N_j 向中央协调节点 N_0 发送一个消息,告诉 N_0 发生了约束失效,需要进行调整。这条消息要包含以下内容。

1) 约束失效的原因:也就是 2.2节介绍的两个原因。

2) 失效约束相关的对象标识 O_i 和此次的变化量 Δ。

3) 和失效约束相关的 $V_{i,j}$ 和 $\delta_{i,j}$,以及 N_j 所知道的在 top-k 集合中的 $V_{x,j}$ 和 $\delta_{x,j}$。

4) 边界值以及边界对象 O_b:如果是原因 1,则发送 G_j,否则发送 H_j。

阶段 2　N_0 判断约束失效的原因,如果是原因 1,则表示 O_i 可能是竞争者,则调用 competitor算法,否则表示 O_i 可能是退出者,调用 quitter算法。

阶段 3　N_0 向所有监视节点请求 $V_{i,j}$ 和边界值,重新初始化 $k'=k$,并据此计算新的 top-k 集合 T 以及调整量,并向各个监视节点发送新的 T 以及各自相关的调整量,各监视节点计算各自的 G_j 和 H_j。下面分别对 competitor算法和 quitter算法进行详细的介绍。

算法 2　competitor算法

Begin
1)　　If $(\forall O_t \in T, \forall O_s \in (U-T),$
$V_{t,j}+\delta_{t,0}+\delta_{t,j}+\varepsilon \geq V_{s,j}+\delta_{s,0}+\delta_{s,j})$
2)　　重算 N_0 和 N_j 的调整因子,并向 N_j 发送它的调整量;
3)　　else
4)　　if $(K' < K_{max})$
5)　　$K'=K'+1$;
6)　　把 O_j 加入到 top-k 集合 T 中;
7)　　通知各监视节点 O_j 加入了 top-k 集合
8)　　else
9)　　进入阶段 3
End

competitor算法首先检测这个约束失效是否会导致当前 top-k 集合的变化,如果不会引起变化,重算 N_0 和 N_j 的调整因子,并向 N_j 发送它的调整因子,以便以后发生变化时能够方便地判断 O_i 是否是竞争者。否则,若当前的 $k' < k_{max}$,只需要把 O_i 加入到 top-k 中,并把 k 增加 1;否则,说明当前 O_i 成了名副其实的竞争者,要把 k 中的一个挤出去,这时候进入第三阶段进行调整。

quitter算法与 competitor算法大致思路相同,不再赘述。

算法 3　quitter算法

Begin
1)　　If $(\forall O_t \in T, \forall O_s \in (U-T),$
$V_{t,j}+\delta_{t,0}+\delta_{t,j}+\varepsilon \geq V_{s,j}+\delta_{s,0}+\delta_{s,j})$
2)　　重算 N_0 和 N_j 的调整因子,并向 N_j 发送它的调整量;
3)　　else
4)　　if $(K' > K)$
5)　　$K'=K'-1$;
6)　　把 O_i 从 top-k 集合中 T 删除;
7)　　通知各监视节点 O_i 退出了 top-k 集合
8)　　else
9)　　进入阶段 3
End

实际上,通过分析算法可以发现,与文献[1]中的算法相比,这个算法是把重新计算阶段集中起来处理,相当于开了一个大小为 $k_{max}-k$ 的缓冲来把一些本应该到 3阶段做的事情集中起来一次处理。

4　实验结果及讨论

4.1　实验环境及查询实例

下面以引言部分的查询 1和查询 2为例,在北大数据流管理系统 Argus系统上进行实验,Argus由服务器程序、数据客户端程序和查询客户端程序组成。由于目前实现真正的分布式环境还有一定的困难,本文采用多线程模拟分布式环境。

查询 1的 CQL语言表示为:

```
SELECT TOP 20 Object
FROM Log
GROUP BY Object
ORDER BY COUNT(*) DESC
```

查询 2的 CQL语言表示为:

```
SELECT TOP 1 Server
FROM Log
WHERE Region = 3
PRECEDING 900 SECONDS
GROUP BY Server
ORDER BY COUNT(*)
```

其中:查询 1的监测对象是网站的所有网页 $U=\{O_1, O_2, \cdots, O_n\}$,网页热门程度用当天累计被访问次数来衡量。每个提交给第 j 个服务器的对第 i 个网页的访问可以用三元组 $\langle O_i, N_j, 1\rangle$ 表示。查询 2的监测对象则是所有的服务器节点:$U=\{N_1, N_2, \cdots, N_m\}$,服务器负载用过去 15 min内接收到的网页请求来衡量。由于算法支持的 top-k 查询是选择值较大的 k 个对象,因此将负载最小等价为负载的相反数最大。每个提交给第 j 个服务器的访问用三元组 $\langle O_j, N_j, -1\rangle$ 表示,15 min后用三元组 $\langle O_j, N_j, 1\rangle$ 表示该次访问已不在查询考虑的范围。

4.2　实验结果分析及讨论

本实验讨论自适应 top-k 算法对算子执行开销的影响。该算法对查询 1的通信开销基本没有影响,对查询 2的通信开销影响显著如图 2所示。这是因为查询 1中的监测对象值只增不减,top-k 集合始终以竞争的形式变化;查询 2中的数据对象有增有减,自适应 k 算法为 top-k 集合交替进行的竞争和补充提供了弹性的空间。实验分析比较了文献[1]中固定 k 值算法和自适应 top-k 算法在求精确解和带有一定误差的几种情况;从图 2中可看出:采用自适应 top-k 算法无论在求精确解还是带有误差的情况下,网络通信开销均明显低于文献[1]中算法。随着监视节点 w 的增加,即监视节点较多时用文献[1]中方法计算 top-k 的精确解时,网络开销增长很快,在某些场合是无法接受的;在引入一定可接受的误差后,通信开销显著降低。

(下转第 174页)

练集构造决策树模型,通过创建的模型在测试集上进行预测,从而得出分类准确率。实验采用 weka软件实现 D3算法构造决策树模型的过程,产生的模型通过 Supplied test set法验证其准确率,其余参数为 weka的缺省值。AD3算法通过 VC++开发工具来实现。实验结果如表 2和表 3所示。

表 2 分类准确率的实验结果

训练集	测试集	AD3分类准确率	D3分类准确率
12	12	0.75	0.75
16	8	0.75	0.50
18	6	0.83	0.67
19	5	0.80	0.60
20	4	1.00	0.50
21	3	1.00	0.33
22	2	0.50	0.50
平均准确率		0.80	0.55

表 3 模型健壮比的实验结果

训练集	测试集	AD3健壮比	D3健壮比
12	12	1.00	1.00
16	8	2.25	2.25
18	6	2.50	2.25
19	5	2.50	2.25
20	4	2.33	2.17
21	3	2.33	2.17
22	2	2.33	2.17
平均健壮比		2.18	2.04

从表 1和表 2的实验结果可以看出,相对 D3算法而言,AD3算法的分类准确率并没有降低,对不同的训练集反而有所提高,平均准确率要高于 D3算法。与 D3算法构造的决策树模型相比,AD3在“根节点数量”、“叶节点数量”、“树的高度”和“规则数量”等方面与 D3有一定的相似性,比 D3算法构造的决策树模型要健壮。实验结果充分说明了 AD3决策树模型不仅克服了多值偏向的问题,而且提高了分类准确率和健壮性。

5 结语

D3是一个经典的决策树算法,对 D3算法固有缺点的改进是有价值的。真实世界的数据一般不可能是完美的,或缺失,或不准确,剪枝是一种克服噪声的技术[5],这些将成为下一步研究的重点。

参考文献:

[1] CHEN JR-SHIAN, CHENG CHING-HSUE. Extracting classification rule of software diagosis using modified MEPA[J]. Expert Systems with Applications: An International Journal, 2008, 34(1): 411 - 418.

[2] 刘鹏,姚正,尹俊杰. 一种有效的 C4.5改进模型[J]. 清华大学学报:自然科学版, 2006, 46(S1): 996 - 1000.

[3] 高学东, 尹阿东,宫雨,等. 一种改进的决策树算法[J]. 工业工程与管理, 2004, (4): 93 - 97.

[4] 韩松来,张辉,周华平. 关于关联度函数的决策树分类算法[J]. 计算机应用, 2005, 25(11): 2655 - 2657.

[5] 卜亚杰,胡朝举,董娜,等. 一种基于 MMEPA的决策树构造方法[J]. 电脑知识与技术, 2007, 1(5): 1292 - 1293.

(上接第 171页)

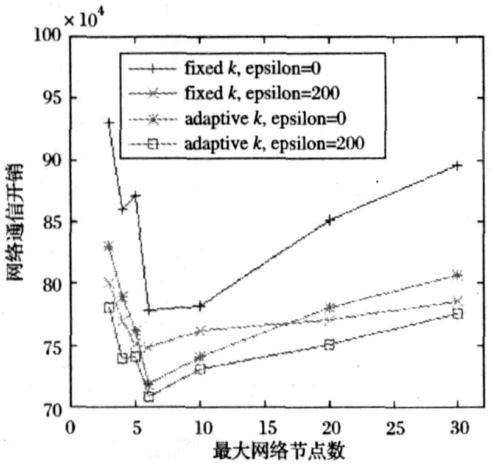

图 2 自适应 top-k算法的性能试验

在实验中发现: k_{max}的选择对通信开销影响不大,这一点与数据的变化特征有关。如果一段时期内存在某个 $k' \in [k, k_{max}]$所对应的 top-k集合较为稳定,自适应 k算法可以保证这段时间内的通信开销保持在很低的水平。而因为数据流的变化特征也处在变化当中,事先指定的 k_{max}就无法保证总让自适应 k 算法有效。

5 结语

对于分布式算法来说,网络通信的时间代价和本地计算代价相距甚远,通信开销就成为衡量算法优劣的重要标准。通过实验表明在分布式数据流环境下, top-k算法虽然引入了一定的误差,但显著地降低了通信开销,而且引入的误差在许多应用场景中是可以接受的。所以说 top-k算法极具应用价值,若在经过经一步的完善,做到根据数据流特征自适应地选择 k_{max}的值,还可使算法执行效率有更大的提高。

参考文献:

[1] BABCOCK B, OLSTON C. Distributed top-k monitoring[C]// ACM SIGMOD. NewYork: ACM Press, 2003: 28 - 39.

[2] The Internet Traffic Archive[EB/OL]. [2005 - 03 - 10]. http://ita.ee.lbl.gov/index.html.

[3] YI K, YU H, YANG J. Efficient maintenance of materialized top-k views[R]. Providence: Department of Computer Science Duke University, 2003.

[4] MOURATIDIS K, BAKIRAS S, PAPADIAS D. Continuous monitoring of top-k queries over sliding Windows[C]// ACM SIGMOD. Chicago, Illinois, USA: ACM Press, 2006: 635 - 646.

[5] DAS G, GUNOPULOS D, KOUDAS N, *et al*. Answering top-k queries using views[C]// The 32nd International Conference on Very large data bases. Seoul, Korea: ACM Press, 2006: 451 - 462.

[6] FAGIN R, LOTEM A, NAOR M. Optimal aggregation algorithms for middleware[J]. Journal of Computer and System Sciences, 2003 (66): 614 - 656.

[7] CAO P, WANG Z. Efficient top-k query calculation in distributed networks[R]. Providence: Department of Computer Science University of California, 2004.