基于不确定数据的分布式 Top-k 查询算法

王 爽[1,2], 王国仁[2]
(1. 东北大学 软件学院, 辽宁 沈阳 110004; 2. 东北大学 信息科学与工程学院, 辽宁 沈阳 110004)

摘 要: 目前基于不确定数据的 Top-k 查询算法仅考虑了集中式的环境,为了解决分布式系统中节省系统带宽的问题,在此基础上,提出了在分布式环境中基于不确定数据的 Top-k 查询算法 UDTopk。该算法定义了一个候选集(candidate set),仅使用候选集中的数据,而不用访问数据集中所有数据,就可以得到正确的 Top-k 查询答案。算法通过动态维护候选集、仅传输少量数据,达到减少网络中数据传输的目的。实验结果表明,该算法可以有效地节省网络带宽。

关 键 词: Top-k 查询; 不确定数据; 分布式处理; 通信代价; 查询处理
中图分类号: TP 393 文献标志码: A 文章编号: 1005-3026(2010)02-0177-04

Distributed Top-k Query Algorithm Based on Uncertain Data

WANG Shuang[1,2], *WANG Guo-ren*[2]
(1. School of Software, Northeastern University, Shenyang 110004, China; 2. School of Information Science & Engineering, Northeastern University, Shenyang 110004, China. Correspondent: WANG Shuang, E-mail: wangshuang_neu@163.com)

Abstract: Top-k query based on uncertain data has quickly attracted a lot of interested users, however, none of them has addressed himself to that the algorithm works in a distributed setting. A distributed Top-k algorithm based on uncertain data(UDTopk) is therefore presented to save the communication bandwidths. A data structure called candidate set is designed and proposed, where only the minimum amount of data is contained and the tuples that have been removed from the set will not affect the answer to a Top-k query. This algorithm presented can be dynamically maintained with new tuples being added, and only small amount of data is required to transmit, thus reducing the data transmission in the network. The experimental results showed that the UDTopk algorithm can effectively reduce the communication cost.

Key words: Top-k query; uncertain data; distributed processing; communication cost; query processing

数据的不确定性普遍存在,其存在性未知而且各属性值存在误差[1-2]。尽管数据预处理能够提升原始数据集合的质量,但也可能会丧失原始数据集合的部分性质,导致无法返回高质量的查询结果[3]。目前针对不确定数据 Top-k 查询存在多种定义方法,例如 U-Topk[4], U-kRanks[4], PT-k[5] 和 Pk-topk[6] 查询等。但这些针对不确定数据的 Top-k 查询算法只考虑了集中式的环境,并没有考虑分布式处理问题。在实际应用中,绝大多数系统采用的都是分布式的体系结构。在分布式环境中,系统的通信带宽是一种瓶颈资源[7]。另外,无线传感器网络应用环境中传感器节点发送数据时消耗的电能也是一种瓶颈资源[8]。因此,通信有效性在分布式数据处理中尤为重要。为此,本文提出了一种基于不确定数据的分布式 Top-k 查询算法 UDTopk 算法(distributed Top-k algorithm based on uncertain data),设计了一个候选集(candidate set),该集合仅需要维护最少的数据就可得到查询的结果,从而减少了数据的传输量,节省了网络的带宽。

收稿日期: 2009-06-29
基金项目: 国家自然科学基金资助项目(60873011)。
作者简介: 王 爽(1980—),女,辽宁沈阳人,东北大学博士研究生; 王国仁(1966—),男,湖北崇阳人,东北大学教授,博士生导师。

1 基于不确定数据的分布式 Top-k 查询

在分布式处理系统中,数据分散在多个处理中心,多个处理中心通过合作为用户提供有关系统全局的数据查询。本文采用单层的分布式体系结构,在这种结构中有两种类型的节点:叶子节点(remote sites)和中心节点(central server)。叶子节点接收到达的数据,中心节点接收从叶子节点发送的数据,从而产生查询结果。在这种结构中,叶子节点仅与中心节点通信,叶子节点之间并不通信。

最基本的查询方法是将每个叶子节点的数据全部传送到中心节点,中心节点根据现有的集中式算法得到 Top-k 查询结果,需要的通信代价为 $N=\sum n_p$,其中 n_p 为每个叶子节点数据集的尺寸。显然,该方法传输代价太大,因此,设计一个通信有效的 Top-k 算法,使系统传输的数据量最小,是十分必要的。

1.1 候选集

本文提出了一个候选集(candidate set)的概念,仅使用候选集中的数据,就可以得到正确的 Top-k 查询答案。一般情况下,数据集 D 中都会包含候选集 $CS(D)$,候选集中的数据量远远小于原始的数据集。叶子节点给中心节点传递数据时,不需要传输叶子节点中的所有数据,只需把候选集中的数据传递给中心节点,就可以得到正确的答案,因为不在候选集中的数据肯定不能成为最终的结果。通过该方法可减少数据的传输量。本文采用 U-kRanks[4] 作为不确定数据的 Top-k 定义。

定义1 如果元组 t_i 的打分函数值比元组 t_j 大,即 $f(t_i)>f(t_j)$,则 $t_i \prec_f t_j$。类似地,如果 $f(t_i)<f(t_j)$,则 $t_i \succ_f t_j$。

本文假设每个叶子节点的数据集 D 已按打分函数降序排列,$t_1 \prec_f t_2 \prec_f \cdots \prec_f t_n$。

定义2 用 $r_{i,j}$ 表示 D_i 中出现 j 个元组的概率[4]。根据元组是否存在,可得

$$r_{i,j}=\begin{cases}1, & i=j=0;\\ p(t_i)r_{i-1,j-1}+ & \\ [1-p(t_i)]\,r_{i-1,j}, & i\geqslant j;\\ 0, & i<j。\end{cases}\tag{1}$$

符号 D_i 表示数据集 D 中打分值最大的 i 个元组集合。显然,D_i 中包含的元组数最多为 i 个,因此当 $i<j$ 时,$r_{i,j}=0$。当 $i\geqslant j$ 时,采用递推公式,分为两种情况:第一种,D_{i-1} 中仅包含 $j-1$ 个元组,则第 j 个元组应该出现在 D_i 中;第二种,D_{i-1} 中已经包含了 j 个元组,则第 j 个元组就不应该出现在 D_i 中。从式(1)的定义可以看出,元组 t_i 排在第 j 位的概率为 $p(t_i)r_{i-1,j-1}$。因为数据集 D 按降序排列,t_i 要排在第 j 位,必须保证前面 $i-1$ 个数据 D_{i-1} 集中只出现 $j-1$ 个数据,这样 t_i 才可能排在第 j 位上。

定义3 候选集 $CS(D)$ 是数据集 D 的一个子集,$CS(D)$ 仅包含得到 Top-k 查询结果需要的最少的数据。候选集中的元组需要满足下面两个条件:

① $\forall t_i\in CS(D)$,$\forall t_j\in D$ 但 $t_j\notin CS(D)$,$t_i \prec_f t_j$。

② t_e 是 $CS(D)$ 中排在最后一位的元组,$\forall t_i\in CS(D)$,必须满足下面的不等式[9]:

$$\max_{1\leqslant i\leqslant e} p(t_i)r_{i-1,j-1}\geqslant \max_{0\leqslant l\leqslant j-1} r_{e,l},\ j=1,\cdots,k。\tag{2}$$

其中式(2)是根据 U-kRanks 的定义得到的。其他的 Top-k 定义需要对式(2)作少量的修改。

条件①要求候选集中元组的打分函数比候选集外元组的大。在式(2)中,$p(t_i)r_{i-1,j-1}$ 表示元组 t_i 排在第 j 位的概率。$\max_{1\leqslant i\leqslant e} p(t_i)r_{i-1,j-1}$ 表示候选集中的元组 $\{t_1,\cdots,t_e\}$ 排在第 j 位的最大概率,则具有最大概率值的元组就是 Top-k 查询结果中排在第 j 位的答案。该元组的概率必须不小于式(2)右边的概率。

证明 $\forall t'$,$t'\notin CS(D)$,另 δ_s 表示 $\{t_{e+1},\cdots,t_n\}$ 中出现 s 个元组的概率。则 t' 排在第 j 位的概率为 $p(t')\sum_{l=0}^{j-1} r_{e,l}\delta_{j-1-l}$,其中 $r_{e,l}$ 表示在候选集中仅出现 l 个元组的概率,δ_{j-1-l} 表示在候选集之外出现 $j-1-l$ 个元组的概率。$p(t')$ 与两者的乘积为 t' 排在第 j 位的概率。

因为 $p(t')\leqslant 1$,且 $\sum_{l=0}^{j-1}\delta_{j-1-l}\leqslant 1$,所以

$$p(t')\sum_{l=0}^{j-1} r_{e,l}\delta_{j-1-l}\leqslant\sum_{l=0}^{j-1} r_{e,l}\delta_{j-1-l}\leqslant\max_{0\leqslant l\leqslant j-1} r_{e,l}。$$

从候选集的定义可以看到,Top-k 查询结果肯定在候选集中。在执行查询时,可以仅仅查找候选集中的数据,而不用查询整个数据集,就可以得到答案,查询时间仅为 $O(k\cdot e)$,节省了时间开销。

性质1 t_{new} 是新加入到数据集 D 中的元组,t_e 是数据集 D 的候选集 $CS(D)$ 中的最后一个元组。如果 $t_{new}\succ_f t_e$,则 $CS(D\cup t_{new})=CS(D)$;

否则 $CS(D \cup t_{new}) \subseteq CS(D) \cup t_{new}$。

性质 1 由候选集的定义 3 很容易证明。根据第一个条件，如果新加入的元组比 t_e 的打分函数还要低，则它不满足第一个条件，不会成为结果，不需要考虑。反之，如果它比 t_e 的打分函数大，则有可能成为答案，此时需要重新计算 $CS(D) \cup t_{new}$ 的候选集才能得到正确答案。由于 $CS(D)$ 之外元组的打分函数肯定也比 t_e 小，因此 $CS(D)$ 之外的元组仍然不需要考虑。

1.2　分布式查询算法 UDTopk

根据候选集的定义，提出了基于不确定数据的分布式 Top-k 查询算法 UDTopk。候选集要求数据已经预先排好顺序，但由于是分布式系统，在每个叶子节点上按降序排序，并不能保证全局也是有序的；因此需要保证传递到中心节点的数据是按序排列的，才可以利用候选集得到查询答案。具体算法如下。

算法：UDTopk(中心节点)。

输出：Top-k 查询结果。

① array $D=\varnothing$，array $CS=\varnothing$；

② 获取每个叶子节点的第一个元组，将该元组添加到数组 D 中；

③ 将数组 D 中的元组按打分函数降序排列；

④ 对数组 D 中的元组计算候选集 CS，并得到 CS 中打分最小的元组 t_e；

⑤ 将 t_e 的打分值广播给所有的叶子节点；

⑥ loop {接收叶子节点传递的元组}

　　$CS(D) := CS(CS(D) \cup \{\text{new tuples}\})$

　　如果没有叶子节点传递数据，

　　循环结束；

⑦ 计算 Top-k 查询结果。

算法：UDTopk(叶子节点)。

输入：不确定数据集 D_p，$1 \leqslant p \leqslant m$(叶子节点数)。

① Loop{接收中心节点元组 t_e 的打分值}；

② 如果目前排在首位元组的打分值大于 t_e 的打分值，

　　将该元组发送给中心节点；

　　否则，结束。

初始化阶段：该算法首先获取每一个叶子节点的第一个元组，因为每个叶子节点的数据都按降序排列，则传递给中心节点的数据肯定是每个叶子节点打分值最大的数据。将这些数据按降序排列，得到候选集，并同时得到候选集中打分值最低的元组，初始化完毕。中心节点将该元组的打分值广播给所有叶子节点。根据性质 1，如果某个叶子节点排在第二位的元组的打分值比广播值还低，则该节点的数据都不需要进行传递；反之，将该节点排在第二位的元组传递给中心节点。依此类推，直到所有的叶子节点的数据都比广播值低，算法终止。

下面分析算法的时间和空间复杂度。W 代表候选集的最大尺寸，S 代表中心节点需要排序数据的个数。计算式(1)中的概率需要 $O(k \cdot W)$ 时间。如果从叶子节点传递给中心的数据打分值大于 t_e，则需要重新计算候选集。由于候选集中总是保存着当前已经传递到中心节点中打分值最大的一些元组，此事件发生的概率为 (W/N)，N 是总的数据集的尺寸。因此总共需要花费的时间为 $O(kW^2/N+\lg S)$。$\lg S$ 为对 S 个元组进行排序的时间。算法的空间复杂度为 $O(S+W)$。

2　性能分析

为了验证算法的有效性，本文采用模拟实验对算法的性能进行了深入研究。实验的硬件环境为 Pentium 1.0 GHz CPU，512 MB 内存，操作系统为 Windows XP。采用 Microsoft VC++ 6.0 编程环境开发了模拟测试程序。实验采用合成数据，分别构造了 3 个含有 50 000 个元组的数据集，分别符合平均分布，正态分布 $N(0.5, 0.2)$ 和正态分布 $N(0.9, 0.2)$。对于每一个叶子节点的数据，从总的数据集(50 000 个元组)中随机选取一些元组，将其分布在不同的叶子节点上。

本文主要从候选集的大小、数据通信量两个方面对实验结果作了对比分析。图 1 显示了候选集尺寸随 k，N_{tup}(元组数)变化的情况，其中总的数据量为 50 000 个元组。随着 k 的增加，候选集呈线性增长。正态分布数据集 $N(0.9, 0.2)$ 比其他分布数据集的候选集尺寸小，因为期望为 0.9，因此数据分布相对集中，而打分值较大的数据的分布也相对集中，所以不需要保存更多的数据。候选集的大小与 N 没有直接关系，因为数据集已经按照打分函数排序，候选集的大小只取决于打分值排在前面的那些元组，而与整个数据集大小没有关系。

图 2 显示了通信量随 k，m 变化的情况，通信量以字节为单位，叶子节点的个数为 10。图中 BC(baseline communication)表示的是将叶子节点中所有数据都上传到中心节点的情况，总数据集有 50 000 个元组，需要传递的消息格式为(i, tid,

score, confidence), 前两个字段用 2 个字节表示, 后两个字段用 4 个字节表示, 则总的通信量为 600 KB。随着 k 的变化, 候选集中的数据增加, 中心节点和叶子节点交互的次数也要增加, 因此通信量增加。随着 m 的增加, 中心节点发送给叶子节点的消息个数增加, 因此通信量增大。

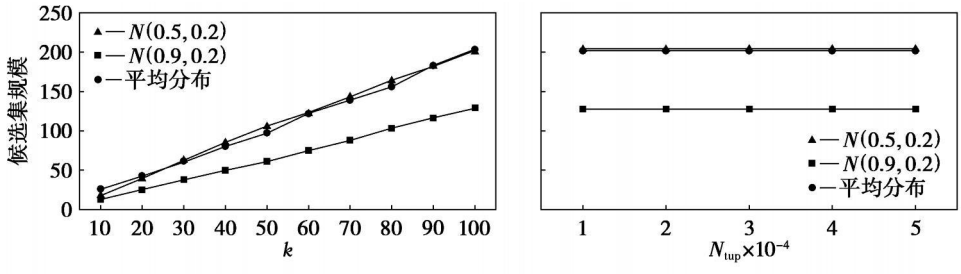

图 1 候选集随 k 和 N_{tup} 变化的情况

Fig. 1 Candidate set size vs. k and N_{tup}

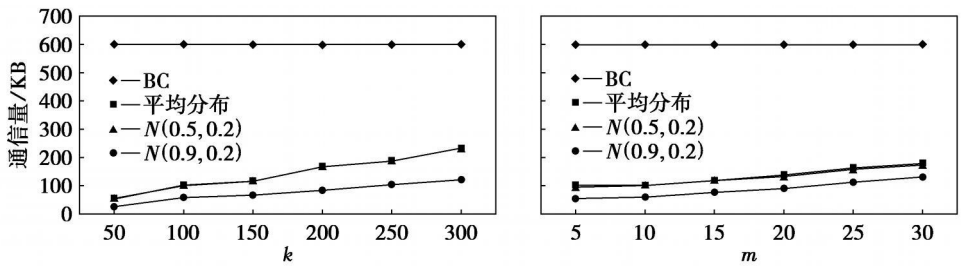

图 2 通信量随 k 和 m 变化情况

Fig. 2 Communication cost vs. k and m

3 结 语

Top-k 查询是数据库领域一种非常重要的查询类型。面向确定性数据库的 Top-k 查询的定义非常清晰, 但针对不确定数据, 需要综合考虑打分函数和概率两种情况。本文基于现有的不确定 Top-k 查询定义, 将其扩展到分布式环境下, 提出了一个通信有效性算法 UDTopk, 通过动态维护一个候选集, 可以有效减少数据通信量, 节省网络带宽。实验结果表明, 该算法可以显著减少数据通信量。

目前本文采用的不确定数据模型相对简单, 仅考虑了元组独立的情况, 而没有考虑元组之间相互依赖的情况。另外, 本文没有考虑数据流的情况。今后需要对该算法进行改进, 使其适用于更复杂的概率模型以及动态数据。

参考文献:

[1] 李建中, 于戈, 周傲英。不确定性数据管理的要求与挑战[J]。计算机学会通讯, 2009, 5(4): 6—14。
(Li Jian-zhong, Yu Ge, Zhou Ao-ying. The requirements and challenges of uncertain data management [J]. *Communications of the CCF*, 2009, 5(4): 6—14.)

[2] Green T J, Tannen V. Models for incomplete and probabilistic information[J]. *IEEE Data Engineering Bulletin*, 2006, 29(1): 17—24.

[3] 周傲英, 金澈清, 王国仁, 等。不确定性数据管理技术研究综述[J]。计算机学报, 2009, 32(1): 1—16。
(Zhou Ao-ying, Jin Che-qing, Wang Guo-ren, *et al*. A survey on the management of uncertain data[J]. *Chinese Journal of Computers*, 2009, 32(1): 1—16.)

[4] Soliman M A, Ilyas I F, Chang K C C. Top-k query processing in uncertain databases[C/OL] // International Conference on Data Engineering. Turkey, 2007[2009—02—15]. http://www.forward.cs.uiuc.edu/pubs/2007/utopk-icde07-sic-nov06.pdf.

[5] Hua M, Pei J, Zhang W J, *et al*. Ranking queries on uncertain data: a probabilistic threshold approach[C/OL] // International Conference on Management of Data. Canada, 2008[2009—03—12]. http://www.cs.sfu.ca/~jpei/publications/uncertaintopk_sigmod08.pdf.

[6] Jin C Q, Yi K, Chen L, *et al*. Sliding window Top-k queries on uncertain streams[C/OL] // Very Large Databases. [2009—01—23]. http://www.cse.ust.hk/~leichen/papers/vldb08-WTOPK.pdf.

[7] Dilmand M, Raz D. Efficient reactive monitoring[C] // IEEE International Conference on Computer Communications. New York: IEEE, 2001: 668—676.

[8] Xue W W, Luo Q, Chen L, *et al*. Contour map matching for event detection in sensor networks[C/OL] // International Conference on Management of Data. [2009—01—17]. http://wireless.cs.uh.edu/presentation/2007/sigmod06.pdf.

[9] Yi K, Li F F, Kollios G. Efficient processing of Top-k queries in uncertain databases with x-relations[J]. *IEEE Trans on Knowlege and Data Engineering*, 2009, 21(1): 1—14.