云环境下的 Max/Min 在线聚集技术研究

汪凤鸣,慈 祥,孟小峰

(中国人民大学 信息学院,北京 100872)
E-mail: wangfengmingqq@163.com

摘 要: 数据探索作为数据分析的一个重要环节,必须能够高效的获取数据集的关键性指标,比如最大/最小值、均值等. 关系数据库中这些指标可以通过 SQL 语句的聚集函数得到. 为了实现海量数据下的高效聚集,关系数据库领域学者提出了在线聚集. 在大数据时代,云环境下的在线聚集技术开始得到重视. 但是目前云环境下的在线聚集研究基本是针对 Count、Sum 等聚集函数,尚未有针对 Max/Min 在线聚集的研究. 本文利用切比雪夫不等式和中心极限定理,通过分位数来衡量 Max/Min 在线聚集的精确度. 实验证明,该方法能够很好的适应大数据环境下的在线聚集,并具有良好的扩展性.

关 键 词: 在线聚集; 云计算; 切比雪夫不等式; 中心极限定理
中图分类号: TP311 **文献标识码:** A **文 章 编 号:** 1000-1220(2015)10-2177-06

Max/Min Online Aggregation in the Cloud

WANG Feng-ming, CI Xiang, MENG Xiao-feng
(School of Information, Renmin University of China, Beijing 100872, China)

Abstract: As an important part of data analysis, data exploration must be able to efficiently access key indicators of data sets, such as max/min, average and etc. These indicators can be obtained by SQL aggregate functions in relational database. In order to achieve this goal in massive dataset, scholars have proposed the concept of onlineaggregation. In the era of big data, online aggregation in the cloud has attracted attentions. Most of the research focuses on the aggregation function such as Count, Sum and other aggregate functions, while there is little works on the Max/Min online aggregation now. In this paper, we use quantile to measure the accuracy of Max/Min online aggregation which induced by chebyshev's inequality and central limit theorem. The experimental results demonstrate the efficiency of the method and it can well adapt to online aggregation for big data.

Key words: online aggregation; cloud computing; chebyshev's inequality; central limit theorem

1 引 言

大数据的出现给传统的数据管理技术带来了巨大的挑战,随着数据量的快速增长,以关系数据库为核心的数据管理和分析技术在很多领域已经无法适用. 数据分析作为大数据价值体现的重要手段得到了广泛关注,在数据分析的前期通常需要利用一个数据探索阶段来掌握数据集的一些关键特性,比如均值、总和、最大/最小值等. 这些操作在传统的关系数据库中可以通过聚集函数来实现,但是随着数据量的急剧增长,一次完整的数据扫描所耗时间很可能是无法接受的. 同时考虑到数据探索的过程往往是多次和反复的,这种使用全表扫描来获取精确聚集值的方法更是不可行. 就实际情况来看,很多时候用户需要的并不一定是完全精确的结果,近似的结果对于其后续的深层次分析已经足够. 在这种情况下,可以考虑使用一些技术手段,通过牺牲数据精度的方式来换取时间上的节约.

关系数据库领域的研究很早就注意到海量数据下聚集查询的问题,并提出了在线聚集的方法来解决海量数据聚集问题,其基本思路正是通过对采样数据的估计来获取符合一定精度的近似查询结果. 近几年来又有不少学者开始研究云环境下的在线聚集,尝试将传统的在线聚集理论应用于云环境下的海量数据,取得了不少成果. 但是这些研究所关注的聚集函数基本集中在 Count、Sum 和 Average,也基本都是利用中心极限定理来处理样本数据. 这些函数的共同特点就是它们度量的都是数据整体的某方面特性. 除了上述几个函数,另外两个最常用的聚集函数 Max 和 Min 却少有人研究,主要原因在于:

1) Max 和 Min 反映的是数据集分布的端点,或者说是极值的特性而不是数据集的整体特性. 随机抽样得到的样本的不确定性可以通过大量样本的累加来消除,但是极值不是一种可累加消除的数据特性. 因此 Max/Min 在线聚集的统计规律非常难估计;

2) 传统的在线聚集,为了衡量准确性,一般会给出一个在一定置信度下的置信区间,以便用户在得到可以接受的置信区间后停止查询. 这个置信区间一般是通过中心极限定理来得到的. 但是 Max/Min 查询所反映的极值特性本身无法用中心极限定理来衡量,因此难以得到一个可信的置信区间. 这

收稿日期: 2014-07-21 收修改稿日期: 2014-09-09 基金项目: 国家自然科学基金项目(61379050,91224008) 资助; 国家"八六三"高技术研究发展计划项目(2013AA013204) 资助; 高等学校博士学科点专项科研基金课题项目(20130004130001) 资助; 中国人民大学科学研究基金项目(11XNL010) 资助. 作者简介: 汪凤鸣,女,1991 年生,硕士研究生,研究方向为云数据管理; 慈 祥,男,1986 年生,博士研究生,研究方向为云数据管理; 孟小峰,男,1964 年生,博士,教授,研究方向为互联网络与移动数据管理.

一点从直观上也很好理解. 基于上述考虑,本文提出直接用样本的最大/最小值作为总体的最大/最小值的估计. 为了衡量该估计的准确性,利用切比雪夫不等式给出当前样本最大/最小值在真实总体中的分位数的下(上)界,同时结合中心极限定理修正估计所产生的误差. 最后在云环境下实现 Max/Min 在线聚集.

2 相关工作

在线聚集最早由文献[1]提出,主要关注关系数据库中单表在线聚集的实现问题. 随后该问题在关系数据库领域得到了一定程度的研究. 文献[2]对文献[1]的工作进行了扩展,提供了基于大样本的置信区间和确定性置信区间的计算方法. 针对多表连接的在线聚集,文献[3]给出了一系列波纹连接(Ripple join)算法. 波纹连接基于离线查询处理中的嵌套连接和哈希连接设计,其目的是在保证增量计算的前提下尽快得到估计结果. 文献[4]通过并行化采样过程和查询处理过程对波纹连接算法进行了改进,提高了置信区间的收敛速度. 然而,当总体数据的分布情况无法得到或者内存溢出时,该算法无法给出具有统计意义的置信区间. 为了解决该问题,文献[5]将传统的排序-合并连接算法同波纹连接算法进行结合,并在查询处理过程中增加了一个收缩的处理模块用于更新估计结果. 文献[6]将在线聚集问题扩展到分布式环境中,并给出相应的统计计算方法.

上述研究工作均在关系数据库领域进行,在云计算环境中的在线聚集实现技术目前也有不少相关工作. HOP(Hadoop Online Prototype)[7]将 Hadoop 中的 MapReduce 处理过程流水线化,允许消费操作在生产操作完成之前对已有的数据进行处理. HOP 能够在 MapReduce 作业执行过程中不断提供数据处理结果的快照,并通过作业的执行进度直接对快照进行扩展来实现对聚集结果的估计,但是没有提供结果的置信区间. COLA[8,9]基于 HOP 的流水线处理技术进行了改进,允许不同任务采用不同的流水线粒度. 并且实现了置信区间的估计,能完成单表和多表的在线聚集. 文献[10]提出了一种基于贝叶斯理论实现在线聚集的方法,该方法考虑每个数据块的聚集值和该数据块处理时间的关系,将数据块的聚集值及其调度时间和处理时间一起进行统计建模. 该方法假设数据块的处理时间越长,其聚集值也越大,但这个假设并不是在所有的聚集操作中均成立. 除此以外,它只解决了由一个 MapReduce 作业构成的单表在线聚集问题,而没有考虑基于多个 MapReduce 作业的多表连接在线聚集实现. 一般而言,随机性越好的数据,其在线聚集的效果就越好. 但是实际的数据常常并不那么理想. 文献[11]针对数据倾斜(data skew)的问题,在整个操作进行之前引入一个预处理阶段,将原本随机性一般的数据变为随机性较好的数据. 由于这个过程是在内存中进行的,效率较高. 文献[12]考虑到多个在线聚集查询同时进行时,抽样样本等系统资源可以在不同查询之间进行共享. 基于此种考虑,文献[12]在具有倾斜分布的数据集上设计和实现了资源共享的多并发查询系统.

无论是在关系数据库中,还是在云环境下,上述这些工作所关注的聚集函数基本集中在 Count、Sum 和 Average. 目前尚未见到云环境下的 Max/Min 在线聚集工作,与本文研究主题最相关的工作是文献[13],文中作者利用贝叶斯方法从样本来推断数据集的极值. 但是该方法本身不是针对在线聚集设计的,且在单机环境下实现,具体的方法也较为复杂.

3 Max/Min 在线聚集的关键问题

中心极限定理(Central Limit Theorem)是统计学中最重要的定理之一,它表明大量相互独立的随机变量,其均值的分布以正态分布为极限. 针对 Sum、Count 等函数的在线聚集正是通过采样的方法,对样本进行统计,然后利用中心极限定理将样本的结果推广到总体,并以一定置信度下的置信区间来衡量估计结果的准确性. 但是通过中心极限定理无法从样本中计算出总体的最大/最小值,也无法给出一个能保证总体最大/最小值所在的置信区间. 因此需要通过别的方法来衡量最大/最小值的估计误差. 本节对 Max/Min 在线聚集的问题进行定义,并给出相应的解决方案.

3.1 问题定义

本文主要关注单表的在线聚集,因此考虑如下的标准聚集查询:

SELECT $op(exp(t_{ij}))$, col FROM R
WHERE $predicate$ GROUP BY col

假设操作的数据表为 R,则在上述查询语句中,op 代表聚集类型(本文中指 Max 或 Min),exp 代表对表 R 的属性进行的代数操作,$predicate$ 是对 R 的元组进行过滤的选择谓词,col 是 R 中一个或者多个属性.

所谓云环境下的 Max/Min 在线聚集就是当用户发出上述标准聚集查询后,利用抽样的样本来估计总体的最大/最小值,并给出满足一定置信度情况下的误差度量. 随着抽样数据量的增大,误差应当逐渐的减小. 当误差达到用户可以接受的范围时就能够停止查询. 置信度可以根据用户的需求来设置,在统计学上一般认为 95% 是比较合适的置信度,因此默认的置信度取该值.

3.2 基于切比雪夫不等式(chebyshev´s inequality)的 Max/Min 估计

置信区间是一个非常好的衡量误差的方法,能够直观反映数据的范围. 但对于 Max/Min 在线聚集而言,给出这样的一个区间不太现实,也不准确. 那么在无法给出相对精确的 Max/Min 值时,究竟什么样的误差度量方式才是有意义的? 询问自己成绩时,你也许会被告知考了 90 分,班里分数超过你的人不会多于总人数的 5%. 在统计某产品销量时,可能会发现该产品的销量处于所有产品的前 1%. 这些近似的统计方式实际上就是统计学中分位数的概念. 分位数是统计学中一个常用的衡量指标,若概率 $0 < p < 1$,则随机变量 X 的分位数 Z_δ 是指满足条件 $p\{X > Z_\delta\} = \delta$ 的实数. 虽然不是全体数据的精确最大(小)值,但是具有很高(低)分位数的值也能在较好程度上反映整体数据在极值上的性质,这就启发我们尝试利用分位数的概念来度量估计值的准确性.

在此思想指导下,通过对样本进行完全随机采样,利用当前采样的最大(小)值作为总体的最大(小)值估计,通过样本的不断增多反复修正结果. 关键的问题是如何得到估计值在真正总体中所处的分位数. 对此,我们利用了统计学中一个著名的不等式——切比雪夫不等式,其定义如下:

如果随机变量 X 的期望为 μ,方差是 σ^2,则对任意 $\varepsilon>0$,满足:

$$P\{|X-\mu|\geqslant\varepsilon\}\leqslant\frac{\sigma^2}{\varepsilon^2} \tag{1}$$

式(1)是切比雪夫不等式的标准形式,一般而言该不等式确定的范围不能够被进一步的优化. 为了计算 Max/Min,我们利用其单边形式,如下:

$$\begin{cases}P\{X-\mu\geqslant t\}\leqslant\dfrac{\sigma^2}{\sigma^2+t^2} & t\geqslant 0\\ P\{X-\mu\leqslant t\}\leqslant\dfrac{\sigma^2}{\sigma^2+t^2} & t<0\end{cases} \tag{2}$$

假设样本中的最大值为 M,则 $M-\mu>0$,根据式(2),得到

$$P\{X-\mu\geqslant M-\mu\}\leqslant\frac{\sigma^2}{\sigma^2+(M-\mu)^2} \tag{3}$$

可得

$$P\{X\geqslant M\}\leqslant\frac{\sigma^2}{\sigma^2+(M-\mu)^2} \tag{4}$$

根据前面分位数的定义,可以确定 M 至少是总体的 $1-\frac{\sigma^2}{\sigma^2+(M-\mu)^2}$ 分位数.

同理假设样本中的最小值为 N,则 $N-\mu<0$,根据式(2),得到

$$P\{X-\mu\leqslant N-\mu\}\leqslant\frac{\sigma^2}{\sigma^2+(N-\mu)^2} \tag{5}$$

可得

$$P\{X\leqslant N\}\leqslant\frac{\sigma^2}{\sigma^2+(N-\mu)^2} \tag{6}$$

也即

$$P\{X\geqslant N\}\geqslant 1-\frac{\sigma^2}{\sigma^2+(N-\mu)^2} \tag{7}$$

因此 N 至多是总体的 $\frac{\sigma^2}{\sigma^2+(N-\mu)^2}$ 分位数.

式(4)和(7)成立的条件是总体的真实期望 μ 和方差 σ^2 已知,但实际上这两个参数并不知道,也需要通过样本来估计,这就必然会产生误差,因此在结果中修正误差才能得出最终的估计值.

3.3　误差修正

从式(4)和(7)中可以看出,误差主要来源于期望和方差的估计. 为了满足计算精度,需要保证在一定置信度 δ 下,样本的期望与真实的期望 μ 误差不超过 ε_μ,而样本的方差和真实的方差 σ^2 误差则不超过 ε_{σ^2}. 其中 ε_μ 的计算等价于下式:

$$P\{|\overline{X}-\mu|\leqslant\varepsilon_\mu\}=\delta \tag{8}$$

由式(8)可得,

$$P\left\{\left|\frac{\overline{X}-\mu}{\sqrt{\sigma^2/n}}\right|\leqslant\frac{\varepsilon_\mu}{\sqrt{\sigma^2/n}}\right\}=\delta \tag{9}$$

即

$$P\left\{\left|\frac{\sqrt{n}(\overline{X}-\mu)}{\sigma}\right|\leqslant\frac{\sqrt{n}\varepsilon_\mu}{\sigma}\right\}=\delta \tag{10}$$

根据统计学理论,此处 σ^2 可以用样本方差 $T_{n,2}=\frac{\sum_{i-1}^{n}(X_i-\overline{X})^2}{n-1}$ 来估计,所以

$$P\left\{\left|\frac{\sqrt{n}(\overline{X}-\mu)}{T_{n,2}^{1/2}}\right|\leqslant\frac{\sqrt{n}\varepsilon_\mu}{T_{n,2}^{1/2}}\right\}=\delta \tag{11}$$

根据中心极限定理,

$$P\left\{\left|\frac{\sqrt{n}(\overline{X}-\mu)}{T_{n,2}^{1/2}}\right|\leqslant\frac{\sqrt{n}\varepsilon_\mu}{T_{n,2}^{1/2}}\right\}\approx 2\Phi\left(\frac{\sqrt{n}\varepsilon_\mu}{T_{n,2}^{1/2}}\right)-1 \tag{12}$$

取 Z_δ 为标准正态分布的 $(\delta+1)/2$ 分位数,则

$$\frac{\sqrt{n}\varepsilon_\mu}{T_{n,2}^{1/2}}=Z_\delta \tag{13}$$

所以 $\varepsilon_\mu=\left(\frac{Z_\delta^2 T_{n,2}}{n}\right)^{1/2}$.

类似的,利用多元中心极限定理,可以得到

$$\varepsilon_{\sigma^2}=\left(\frac{Z_\delta^2(T_{n,4}-T_{n,2}^2)}{n}\right)^{1/2}) \tag{14}$$

其中 $T_{n,4}=\frac{\sum_{i-1}^{n}(X_i-\overline{X})^4}{n-1}$

因此在置信度为 δ 时,M 至少是总体的 φ_M 分位数,其中

$$\varphi_M=1-\frac{\sigma^2+\varepsilon_{\sigma^2}}{\sigma^2+\varepsilon_{\sigma^2}+(M-\mu-\varepsilon_\mu)^2} \tag{15}$$

而 N 至多是总体的 φ_N 分位数,其中

$$\varphi_N=\frac{\sigma^2+\varepsilon_{\sigma^2}}{\sigma^2+\varepsilon_{\sigma^2}+(N-\mu-\varepsilon_\mu)^2} \tag{16}$$

4　云环境下的 Max/Min 在线聚集实现

采样是在线聚集最重要的环节之一,不同的采样方法提供了对总体数据不同方式的访问. 传统在线聚集的估计工作都是基于对表中元组的简单随机采样,保证每条元组被采到的概率是相等的. 然而这种基于元组的统一随机采样在云计算环境中的效率不高. 因为在云计算环境中,数据被组织成块进行存储和管理,而且节点间数据传输也以块为单位. 除此以外,MapReduce 中数据处理单元默认情况下也是数据块,在对 Map 任务进行调度时,以数据块为单位来分配任务. 这种情况下如果不引入额外的处理环节,我们能够采样的最细粒度只能是块. 文献[8]证明了在云计算环境中相同的数据传输代价下,基于数据块的统一随机采样所产生的估计结果并不比基于元组的简单随机采样差. 为了实现数据的随机化,以便更准确的估计,我们对数据进行了一个简单的二次随机抽样. 首先对数据块进行随机抽样,然后在读取块中数据时再进行块中数据的抽样. 主要目的是在不增加抽样复杂度的情况下尽可能保证抽样的随机性. 因为相对其他函数的在线聚集,抽样的随机性对 Max/Min 在线聚集的结果精度和收敛时间等的影响更大.

基本步骤如下:

1) 数据采样. 使用上述的简单二次随机抽样来获取查询的数据样本.

2) 根据具体的查询获取样本的最大/最小值,并以此作为当前的最大/最小值估计;

3) 计算样本的均值和方差;

4) 计算均值和方差的误差;

5) 根据切比雪夫不等式计算最大/最小值所在分位数;

6) 输出结果,继续进行数据采样;

7) 重复上述过程,不断更新结果,直到用户主动停止或数据扫描结束.

基于单表的在线聚集需要一个 MapReduce 作业实现. Map 任务的函数设计如算法 1 所示:

算法 1. Map 函数设计

Input: Object *t*;

Output: Text *key*,Text *value*;

```
1:  if t satisfies the predicate then
2:    key.set(t.tuple.lang);
3:    value.set(t.tuple.size);
4:  end if
5:  output.collect(key,value);
```

Reduce 函数的设计如算法 2 所示:

算法 2. Reduce 函数设计

Input: Text *key*,Iterator (Text) *values*;

Output: Max,Min,fi_max,fi_min

```
1:  //size_n: number of tuples processed by the reducer
2:  //sum_i: sum of the variables in the last iteration
3:  //Max: evaluate max
4:  //Min: evaluate min
5:  //fi_max: Max quantile
6:  //fi_min: Min quantile
7:  while values.hasNext() do
8:    Text it = values.getNext();
9:    val + = sum;
10:   t_n2 + = (list.get(i) - avg)^2;
11:   t_n4 + = t_n2^2;
12:   sega2 = t_n2/(size_n - 1);
13:   avg_err = (1.96^2* sega2/size_n)^(1/2);
14:   sega4 = t_n4/(size_n - 1);
15:   dev_err = ((1.96^2* (sega4 - sega2^2))/(size_n))^(1/2);
16:   fi_max = 1 - ((sega2^2 + dev_err^2)/(sega2^2 +
        dev_err^2 + (Max - avg - avg_err)^2));
17:   fi_min = (sega2^2 + dev_err^2)/(sega2^2 +
        dev_err^2 + (Min - avg - avg_err)^2);
18: end while
output.collect(key,new Text(res))
```

5 实验结果与分析

5.1 实验环境和基本设置

硬件环境方面,实验测试平台是一个由 11 个节点构成的云计算环境,节点之间通过 1Gbit 的交换机相连. 其中一个节点作为 HDFS 和 MapReduce 的主节点(master),剩余的 10 个节点作为工作节点(slave). 每个节点拥有 2.33G 的四核 CPU 和 7GB 内存,每个节点的磁盘大小为 1.8TB. HDFS 的块大小设置为 64M.

软件环境方面,传统的 MapReduce 不支持数据的流水化操作,因此无法实现在线聚集. 我们在 COLA 系统上进行改进,实现了本文所描述的 Max/Min 功能模块,并基于此进行实验.

数据方面,我们使用 Wikipedia 网页访问日志[13] 作为基本的实验数据. 压缩后的数据量大小为 320GB(压缩前为 1TB). 我们从原始数据中抽取了大约 100G 的数据作为本文的实验数据集. 对此数据集我们构造了表 visit_log,包括三个列: 网页名称、网页语言和访问次数. 我们通过对访问次数(pageviews)属性的统计来测试本文方法. 没有使用全部数据集的主要原因是在原始数据集中访问次数属性实际可取的最小值为 1,且这样的值非常多. 这些值对 Max 查询不会有影响,但是在进行 Min 查询时很容易导致实验刚开始就得到最小值为 1 且后期不发生任何变化,无法体现本文方法的有效性. 因此从全部数据集中抽取了部分数据来进行本文实验. 所有实验数据文件存储在 HDFS 上. 使用如下查询语句来进行单表的 Max/Min 在线聚集:

```
Q1 = SELECT Max(pageviews),language FROM visit_log
GROUP BY language
Q2 = SELECT Min(pageviews),language FROM visit_log
GROUP BY language
```

系统的初始置信度设置为 0.95,意味着式(13)中的 Z_δ 为标准正态分布的 0.975 分位数. 查询标准正态分布的分位数表可得 $Z_\delta = 1.96$.

5.2 性能测试与分析

我们在处理后的 Wikipedia 数据集上运行上述两个查询. 聚集结果估计值的准确性使用相对错误 relative_error 衡量,relative_error 在查询执行完成后通过真实的聚集结果计算. 计算公式如下:

$$relative\_error = \frac{|estimateValue - actualValue|}{actualValue} \quad (17)$$

分位数的收敛速度通过平均响应时间 avgtime_max 和 avgtime_min 反映,avgtime_max 是在置信度取 0.95 时样本达到 0.99 分位数的平均耗时. avgtime_min 则是置信度取 0.95 时样本达到 0.01 分位数的平均耗时.

表 1 最大值查询分位数变化表

Table 1 Quantile of Max online aggregation

估计值	真实值	分位数
165	3574	0.7288
552	3574	0.8691
552	3574	0.8870
552	3574	0.9170
911	3574	0.9704
…		
1229	3574	0.9769
1441	3574	0.9818
2922	3574	0.9804
3574	3574	0.9935
3574	3574	0.9954
…		

表 2 最小值查询分位数变化表

Table 2 Quantile of Min online aggregation

估计值	真实值	分位数
325	12	0.7241
194	12	0.6571
85	12	0.4593
59	12	0.2069
59	12	0.2128
…		
33	12	0.1002
33	12	0.0994
24	12	0.0226
12	12	0.0124
12	12	0.0135
…		

注: 1. 上述最大和最小值的真实值都是在扫描完全部的数据后得到.
2. 由于篇幅限制,绝大部分中间结果被省略

选择 0.99 和 0.01 分位数除了这两个值在统计学上的意义之外,我们在多次实验过程中也发现,绝大多数情况下当分位数

达到0.99 时估计的最大值就是真实的最大值,在分位数达到0.01 时估计的最小值就是真实的最小值. 上页表 1 和上页表 2 分别展示了某次实验中最大和最小值查询分位数的变化情况.

实验的原始数据集中包含十种不同语言的网页访问日志,由于篇幅限制,实验结果中展示了相对常用的英语和法语两种语言的结果.

5.2.1　结果误差

图 1 表明了查询 Q1 的相对误差随着查询不断进行的变化过程. 从图中可以看出,同 Count、Sum 等在线聚集不同的是,Max/Min 在线聚集在查询的初期误差还是相当高的. 但是随着查询的进行,误差下降的也比较快. 对于图中的英语而言,大约在查询进行到 15% 左右的时候,相对误差已经不超过5% 了. 在30% 左右时已经得到了真实的最大值,误差为0,且不再变化. 法语的变化趋势类似,但是其变化速率略缓,得到真实最大值的时间也比英语稍长.

图2则是两种语言的Q2查询误差的示意图. 从图中可发现,同 Max 在线聚集相比,Min 在线聚集最初的误差相对较小,误差下降速率也比 Max 稍快. 这可能是由于数据集中访问量少的数据比较多,导致该数据容易作为样本被抽中.

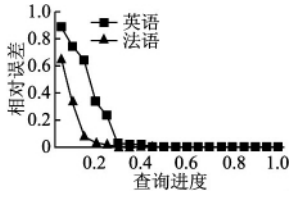

图 1　Q1 查询误差

Fig. 1　Query error of Q1

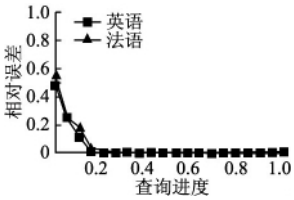

图 2　Q2 查询误差

Fig. 2　Query error of Q2

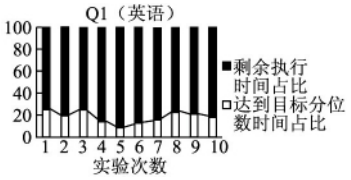

图 3　Q1 查询耗时(英语)

Fig. 3　Query time of Q1(english)

5.2.2　耗时对比

在 11 台机器上对 100G 的数据集重复进行 10 次实验.

图 3 是英语的 Q1 查询耗时情况,图中每个柱状图代表一次实验,下方的柱长代表在本次实验中得到符合精度的结果所耗时间占扫描完全部数据所耗时间的比重. 图中的折线反映了所占时间的变化趋势. 从图 3 中看出,多次查询得到符合精确结果的时间还是相对稳定的.

图 4 反映了法语的 Q1 查询耗时情况,从图中可以看出,法语的查询也比较稳定,但其获得符合结果的时间要稍慢于英语.

图 5 是英语的 Q2 查询耗时情况. 跟图 3 相比较而言,英语的 Q2 查询获取符合精确结果的时间要略快.

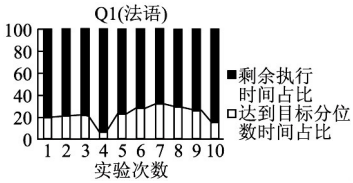

图 4　Q1 查询耗时(法语)

Fig. 4　Query time of Q1(french)

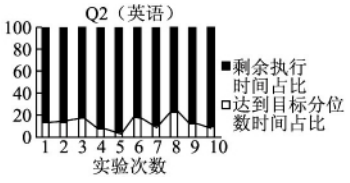

图 5　Q2 查询耗时(英语)

Fig. 5　Query time of Q2(english)

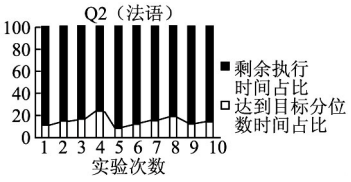

图 6　Q2 查询耗时(法语)

Fig. 6　Query time of Q2(french)

图 6 是法语的 Q2 查询耗时情况. 虽然法语的 Q1 查询要略慢于英语,但是对比图 5 和图 6,发现二者在 Q2 查询上的速率相对接近. 可能的原因还是因为数据集中最小值比较多,在采样时相对容易抽中.

5.2.3　扩展性测试

为了测试本文方法的可扩展性,本文设计了两组实验:

1) 固定集群机器数量,分别对 20G、40G、60G、80G 和 100G 的数据进行 Q1 和 Q2 查询.

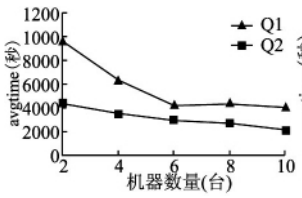

图 7　数据集扩展性测试

Fig. 7　Scalability of data

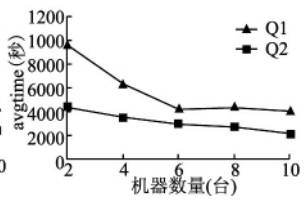

图 8　集群规模扩展性测试

Fig. 8　Scalability of cluster

图 7 展示了随着数据量增大,查询的平均响应时间变化. Q1 的查询整体稳定,但是也有一定的微小波动,总的来看还是比较稳定的. 相比而言 Q2 查询时间更加稳定,但是也并未呈现一种完全线性的增长.

2) 固定数据量为 100G,将集群中 slave 的数量分别设为 2、4、6、8、10. 从图 8 中可以看出,随着机器数量的增加,整体的查询时间基本呈现减少趋势,这说明本文的方法具有较好的可扩展性.

6　小　结

传统的在线聚集研究主要研究 Count、Sum 这类跟数据总体有密切关系的聚集函数. 但是 Max 和 Min 反映的是总体的极端情况,无法用中心极限定理来计算,也不适合用置信区间来衡量数据的精度. 结合实际的应用,发现分位数是一个具有现实意义的精度衡量. 将抽样样本中的最大/最小值作为总体的最大/最小值的估计,并利用切比雪夫不等式和中心极限定理给出该估计在满足一定置信度条件下的分位数. 实验表明了该方法的有效性和扩展性.

但是通过实验结果的分析也发现该方法对于数据集的分布是有一定的要求的,相对标准的分布得到准确结果的时间会非常短,相对不是很标准的数据效果则不是很理想. 因此未来会对现有的工作进行一系列的改进,主要包括: 改进现有的简单随机采

样,设计能够消除分布影响的复杂抽样方法;尝试将最大值的估计扩展到 Top-K 的估计上,以期获得更广泛的应用.

References:

[1] Joseph M Hellerstein, Peter J Hass, Helen J Wang. Online aggregation [C]. Proceedings of ACM Conference on Management of Data, New York: ACM, 1997: 171-182.

[2] Peter J Haas. Large-sample and deterministic confidence intervals for online aggregation [C]. Proceedings of International Conference on Scientific and Statistical DB Management, Piscataway, NJ: IEEE, 1997: 51-63.

[3] Peter J Haas, Joseph M Hellerstein. Ripple joins for online aggregation [C]. Proc of SIGMOD 1999, New York: ACM, 1999: 287-298.

[4] Gang Luo, Curt J Ellmann, Peter J Haas, et al. A scalable hash ripple join algorithm [C]. Proceedings of ACM Conference On Management of Data, New York: ACM, 2005: 252-262.

[5] Chris Jermaine, Alin Dobra, Subramanian Arumugam, et al. A disk-based join with probabilistic guarantees [C]. Proceedings of ACM Conference on Management of Data, New York: ACM, 2005: 563-574.

[6] Wu Sai, Jiang Shou-xu, Beng Chin Ooi, et al. Distributed online aggregation [J]. The Proceedings of the VLDB Endowment, 2009, 2(1): 443-454.

[7] Tyson Condie, Neil Conway, Peter Alvaro, et al. Online aggregation and continuous query support in Mapreduce [C]. Proceedings of ACM Conference on Management of Data, New York: ACM, 2010: 1115-1118.

[8] Shi Ying-jie, Meng Xiao-feng, Wang Fu-sheng, et al. You can stop early with COLA: online processing of aggregate queries in the cloud [C]. Proceedings of ACM International Conference on Information and Knowledge Management, New York: ACM, 2012: 1223-1232.

[9] COLA [EB/OL]. http://idke.ruc.edu.cn/COLA/, 2014.

[10] Niketan Pansare, Vinayak R Borkar, Chris Jermaine, et al. Online aggregation for large mapreduceJobs [J]. The Proceedings of the VLDB Endowment, 2011, 4(11): 1135-1145.

[11] Vasiliki Kalavri, Vaidas Brundza, Vladimir Vlassov. Block sampling: efficient accurate online aggregation in MapReduce [C]. Proc of Cloud Com'13, Piscataway, NJ: IEEE, 2013: 250-257.

[12] Wang Yu-xiang, Luo Jun-zhou, Song Ai-bo, et al. OATS: online aggregation with two-level sharing strategy in cloud [J]. Distributed and Parallel Databases, 2014, 32(1): 1-39.

[13] Wu Ming-xi. Chris Jermaine: Guessing the extreme values in a data set: a Bayesian method and its applications [J]. The VLDB Journal, 2009, 18(2): 571-597.

[14] Wikipedia page traffic statistics [EB/OL]. http://aws.amazon.com/datasets/2596, 2014.