超对等网络中的轮廓查询优化

黄震华[1,2],向 阳[1],孙圣力[3],陈 千[1]
(1.同济大学计算机科学与工程系,上海 201804;2.同济大学嵌入式系统与服务计算教育部重点实验室,上海 201804;
3.北京大学软件与微电子学院,北京 102600)

摘 要: 轮廓查询是近年来信息服务领域的一个研究重点和热点.现有的三阶段算法 TPAOSS (Three-Phase Algorithm for Optimizing Skyline Scalar)至少存在如下两个缺陷:(1)在 TPAOSS 算法的第 3 阶段中,当网络节点上的对象个数较多时,Bloom filter 的长度将呈指数级增长,从而严重影响获取子空间重复值的效率以及占用内存空间的大小;(2) TPAOSS 算法只考虑预处理阶段的时间代价,而没有考虑各网络节点进行局部或全局子空间轮廓查询计算的效率.为此,提出一种适合超对等网络(Super-Peer Architecture,SPA)的子空间轮廓查询方法 EPSSQDN (Efficient Processing of Subspace Skyline Queries in Distributed Networks).EPSSQDN 算法有效解决了 TPAOSS 算法的的两个主要性能问题,并且显著提高了 SPA 网络中的子空间轮廓查询处理的效率.此外,为了能够进一步降低子空间上轮廓查询的时间开销以及网络节点间的数据传输量,我们给出新颖且有效的优化策略.实验结果表明,EPSSQDN 算法比 TPAOSS 算法更能够缩短 SPA 网络中子空间轮廓查询的时间开销.

关键词: 轮廓查询; SUPER-PEER 体系架构; 信息服务; 查询优化
中图分类号: TP311.13 **文献标识码:** A **文章编号:** 0372-2112 (2013) 08-1515-06
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2013.08.010

Optimizing Skyline Queries in SPA Distributed Networks

HUANG Zhen-hua[1,2], XIANG Yang[1], SUN Sheng-li[3], CHEN Qian[1]
(1. *Department of Computer and Technology, Tongji University, Shanghai* 201804, *China*;
2. *The Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai* 201804, *China*;
3. *School of Software and Microelectronics, Peking University, Beijing* 102600, *China*)

Abstract: Skyline query has recently received a lot of attention in information service community. The TPAOSS (Three-Phase Algorithm for Optimizing Skyline Scalar) algorithm has two performance drawbacks: (1) in the third phase of TPAOSS, as the number of objects on net nodes increases, the length of bloom filter will increases exponentially, which will seriously influence the efficiency of obtaining replicated values and the occupation size of memory; (2) the TPAOSS algorithm does not consider the computation efficiency of local or global subspace skyline queries in each net node. Motivated by these facts, we propose EPSSQDN (Efficient Processing of Subspace Skyline Queries in Distributed Networks), an algorithm for efficient processing of subspace skyline queries in SPA distributed networks. Moreover, in order to further reduce the computation cost of subspace skyline queries and decrease the volume of data transferred, we present an efficient optimized techniques. Furthermore, we present extensive experiments that demonstrate our method is more advantageous than the TPAOSS algorithm.

Key words: skyline query; SUPER-PEER architecture; information service; query optimization

1 引言

近年来,轮廓查询技术成为信息服务领域的一个研究重点和热点[1].在现实应用中,用户总共可提交 2^k-1 个不同维空间上的轮廓查询,其中每个维空间对应一个维度组合.依据文献[2]的定义,我们称所有 k 个维度组成的空间为全空间(即 F),而把其余的维空间称为子空间.

随着分布式网络的深入应用,文献[3~6]开始研究在不同架构的分布式网络中,如何有效进行全空间轮廓查询.其中文献[3]、文献[4]、文献[5]和文献[6]分别针对 CAN[7]、SSW[8]、BATON[9]和 CHORD[10]架构.然而值得

收稿日期:2012-02-25;修回日期:2013-01-31;责任编辑:覃怀银
基金项目:国家自然科学基金(No.61272268,No.71171148);教育部新世纪优秀人才支持计划(No.NCET-12-0413);同济大学中央高校基本科研业务费专项资金;江苏省自然科学基金(No.BK2010139)

注意的是,文献[3~7]均只针对全空间,而不考虑任意一个子空间的情况,这是不现实的.为了更具实用性,文献[2]考虑在 SUPER-PEER 体系架构(SPA)[11]的分布式网络中进行任意子空间轮廓查询.SPA 架构是目前使用最广的一种分布式网络,因为传统 C/S 模型的网络架构能够方便地升级为 SPA 架构的分布式网络.在 SPA 网络中,核心是由几台处理能力较强的计算机(称为超点计算机)组成,每台超点计算机关联若干台性能较差的计算机(称为简单计算机),而这些简单计算机为整个网络的终端节点.通常情况下,超点计算机的数量远远小于简单计算机的数量;同时,被查询的数据水平分割存储于这两类计算机上;并且子空间轮廓计算的工作主要在超点计算机上完成.当用户在某一台超点计算机 Initiator 发出子空间 V 上的轮廓查询指令 SKYQ(V)时,查询处理系统通过如下 4 个基本步骤返回给用户所有正确的结果:(1)Initiator 计算机将 SKYQ(V)指令在网络中层层传播;(2)每台简单计算机将相关的数据通过网络传送给与它关联的超点计算机;(3)每台超点计算机接收到所有数据后首先执行子空间 V 上的轮廓计算,并获取相应的计算结果,然后将得到的计算结果通过网络传送给 Initiator;(4)Initiator 接收到所有从超点计算机传送来的数据后首先与它上面的数据合并,并对合并后的数据再次执行 V 上的轮廓计算,然后将所得到的计算结果返回给用户.我们把步骤(1)、(2)、(3)和(4)分别称为指令传播阶段、预处理阶段、局部查询计算阶段以及全局查询计算阶段.

虽然文献[2]给出了一个能够降低网络传输代价的三阶段算法 TPAOSS (Three-Phase Algorithm for Optimizing Skyline Scalar)来处理多个子空间上的轮廓查询.然而,据我们分析,TPAOSS 算法至少存在两个重要性能缺陷:①在 TPAOSS 算法的第 3 阶段中,当网络节点上的对象个数较多时,Bloom Filter[2]的长度将呈指数级增长,从而严重影响获取子空间重复值的效率以及占用内存空间的大小;②TPAOSS 算法只考虑预处理阶段的时间代价,而没有考虑各网络节点进行局部或全局子空间轮廓查询计算的效率.为此,本文着重解决如下问题:如何在 SPA 网络中,给出有效算法 EPSSQDN (Efficient Processing of Subspace Skyline Queries in Distributed Networks)来回答用户发出的任意子空间轮廓查询.包括 3 个方面主要工作:

(1)给出一个复杂度为 $O(|\mathrm{AD}|(\log|\mathrm{AD}|+|V|(|\mathrm{AD}|)$的启发式方法 AOBF(Algorithm for Optimizing Bloom Filter)来提高子空间重复值的获取效率和缩减查询占用内存的空间,其中 AD 为数据对象集合、V 子空间的维数.

(2)提出有效进行任意子空间轮廓计算的方法 ASSC (Algorithm for Subspace Skyline Computation).ASSC 方法基本正规格结构[12]来索引超点计算机上的数据对象,并且使用格间的支配关系来有效降低数据对象间的比较次数,从而提高任意子空间轮廓计算的效率.

(3)提出有效的优化策略来缩短子空间轮廓查询的代价.该优化策略用于缩减超点计算机向 Initiator 传输的数据量与降低局部查询计算的时间开销.

2 AOBF 启发式方法

在文献[2]的 TPAOSS 算法中,简单计算机 SC 直接为 AD-S_f 指定一个长度为 $8\times|\mathrm{AD}\text{-}S_f|$ 的一维数组作为 Bloom Filter,其中 AD 和 S_f 分别为数据对象集合和全空间轮廓集合.然而,我们发现,这种指定方式使得当网络节点上的对象个数较多时,Bloom Filter 的长度将呈指数级增长,从而导致子空间重复值的获取效率极其低下,并且浪费巨大的内存空间.

由于通过扩展支配关系 $\lhd_F$ 得到的全空间扩展轮廓集合 $\Delta^F(\mathrm{AD})$[2]比原始数据集 AD 小,因此当 SC 构造 Bloom Filter 时,我们只考虑 $\Delta^F(\mathrm{AD})\text{-}S_f$ 中的对象即可,而不必考虑 AD-S_f 中的对象.在算法中,SC 必须能够同时算出 $\Delta^F(\mathrm{AD})$和 S_f 这两个数据集,并将 S_f 从 $\Delta^F(\mathrm{AD})$中分离.为此,我们给出一个时间复杂度为 $O(|\mathrm{AD}|(\log|\mathrm{AD}|+|V|(|\mathrm{AD}|)$的启发式算法 AOBF 来完成这个工作.AOBF 方法的伪代码见算法 1 所示.

算法 1 AOBF 方法

输入:k 维对象集合 AD.

输出:全空间轮廓集合 S_f,全空间扩展轮廓集合与 S_f 的差集 S_e.

```
Begin
1.  S_f ← ∅;  S_e ← ∅;
2.  将 AD 划分为 k 个子集,其中第 i 个子集 AD_i 中对象 p 具有如下特征: p[i] = min_{t=1}^{k} p[t];
3.  for i = 1 to k do
4.      flag_i ← True;
        /* flag_i 标识 AD_i 是否需要继续处理 */
5.      对 AD_i 中的对象按第 i 维值非递减排序;
6.  mn ← min_{i=1}^{k} {α_i[i]};
    mx ← min_{i=1}^{k} {max_{t=1}^{k}(α_i[t])};
    /* α_i 为 AD_i 的第一个元素 */
7.  for i = 1 to k do
8.      if mx < α_i[i] then flag_i ← false;
9.  while 存在某子集 AD_j,其标识符 flag_j = True do
10.     S_temp ← ∅;  S_pf ← ∅;  S_ns ← ∅;  L ← ∅;
11.     for i = 1 to d do
12.         L ← {r | r[i] = mn};
13.         mx ← min{mx, max_{t=1}^{k}(r[t])}
14.         将 r 从它所在的数据子集中删除;
15.     S_ns ← {a ∈ L | ∃ x ∈ L, a ≺_F x};  S_pf ← L − S_ns;
16.     S_temp ← {w ∈ S_pf | ∃ y ∈ S_f, w ≺_F y};
```

$S_{ns} \leftarrow S_{ns} \cup S_{temp}$; $S_{pf} \leftarrow S_{pf} - S_{temp}$;

17. $S_{ns} \leftarrow S_{ns} - \{e \in S_{ns} \mid \exists z \in S_f, e \lhd_p z\}$;
$S_e \leftarrow S_e \cup S_{ns}$; $S_f \leftarrow S_f \cup S_{pf}$;

18. $mn \leftarrow \min_{i=1}^{u}\{\beta_i[t]\}$;
/* u 为目前标识为 True 的子集个数,而 β_i 表示第 i 个子集的首个对象 */

19. for $i = 1$ to u do

20. if $mx < \beta_i[i]$ then $flag_i \leftarrow$ false;

21. return S_f, S_e;

End

AOBF 方法首先将 AD 分为 k 个子集,其中第 i 个子集 AD_i 中的每个对象 p 具有如下特点:$p[i] = \min_{t=1}^{k} p[t]$;并对 AD_i 中的对象按第 i 维的值非递减排序.然后,AOBF 方法定位每个子集 AD_i 的第一个对象 α_i,计算出 $mn = \min_{i=1}^{k}\{\alpha_i[i]\}$ 以及 $mx = \min_{i=1}^{k}\{\max_{t=1}^{k}(\alpha_i[t])\}$ 这两个值,并删除所有满足条件 $mx < \alpha_j[j]$ 的子集 AD_j.接下来,AOBF 方法初始化 S_f 和 S_e 为空,其中 S_f 存放最终的全空间轮廓对象,而 S_e 存放所有属于全空间扩展轮廓集合但不属于全空间轮廓集合的对象.并且重复如下过程直到所有 k 个子集被全部删除为止:对于每个保留下来的子集 AD_w,AOBF 方法获取 AD_w 中所有满足如下条件的对象 r:$r[w] = mn$,并修改 mx 的值为 $\min\{mx, \max_{t=1}^{k}(r[t])\}$,同时将这些对象放于列表 L 中.当将所有满足条件的对象放进 L 后,AOBF 方法将那些在全空间上被 L 中其它对象支配的对象添加进集合 S_{ns}中,并将保留下来的对象添加进 S_{pf}中.之后,AOBF 方法将 S_{pf}中在全空间上被 S_f 集合里面的对象支配的对象添加进 S_{ns}中,并从 S_{ns}中删除那些在全空间上被 S_f 集合里面的对象扩展支配的对象,并将保留下来的对象添加进 S_e 中,同时将 S_{pf}中保留下来的对象添加进 S_f 中.最后,更新 mn 的值为 $\min_{i=1}^{u}\{\beta_i[t]\}$,其中 u 为目前所剩余的子集个数,而 β_i 表示第 i 个子集的首个对象,并删除所有满足条件 $mx < \alpha_j[j]$ 的子集 AD_j.

定理 1　(AOBF 方法的正确性)　AOBF 方法能够正确产生全空间轮廓集合 S_f,以及全空间扩展轮廓集合与全空间轮廓集合的差集 S_e.

证明　由于在算法的 while 循环中,我们每次获取的对象具有 $r[i] = mn$ 特点,所以 L 中的每个对象不可能在全空间上被它里面的其它对象所扩展支配,因此,AOBF 方法能够保证 S_{pf}是 L 上的全空间轮廓集合,而 S_{ns}是 L 上的全空间扩展轮廓集合与全空间轮廓集合的差集.同时,ODESP 算法第 16 步能够保证从 S_{pf}中过滤掉那些在全空间上被 S_f 支配的对象,并且将这些对象转移到 S_{ns}中,而算法第 17 步能够保证从 S_{ne}中过滤掉那些在全空间上被 S_f 扩展支配的对象.因此,经过这两步之后,S_{pf}成为 $\nabla^F(AD)$的一个子集,而 S_{ns}成为 $\Delta^F(AD) - \nabla^F(AD)$的一个子集.另一方面,由于 AD 划分得到的每个子集 AD_i,AOBF 方法对 AD_i 中的对象按第 i 维的值非递减排序,所以算法中的 $mx < \alpha_i[i]$或 $mx < \beta_i[i]$能够保证删掉的子集中的对象不可能是全空间扩展轮廓对象,因此,它们也不可能是全空间轮廓对象.从而使得 AOBF 方法返回的 S_f 即为正确的全空间轮廓集合,而 S_e 为正确的全空间扩展轮廓集合与全空间轮廓集合的差集.

3　超点计算机上子空间轮廓计算的 ASSC 方法

局部查询计算阶段和全局查询计算阶段均涉及到任意子空间轮廓计算问题.所不同的是,在局部查询计算阶段,我们完成从简单计算机传来的 S_f 中获取种子轮廓集合的工作;而在全局查询计算阶段,我们完成从各超点计算机传来的局部轮廓集合中获取最终的全局轮廓集合的工作.注意,只有用户发出查询所在的超点计算机(Initiator)必须同时完成局部查询计算阶段以及全局查询计算阶段的工作,而其余的超点计算机只需完成局部查询计算阶段的工作.

为了使 ASSC 方法能够有效处理超点计算机上任意子空间轮廓查询,本文使用正规格结构[12]作为其索引结构.

已知一个单元格的位置,在正规格索引中,我们很容易就可以计算出该格所包含对象的各维取值范围.例如,对于 k 维的格索引 G,G 中的格 $c(wz_1, \cdots, wz_d)$所包含的对象的第 $j(j \in [1, k])$个维度的取值范围为 $((wz_j - 1)\cdot\delta, wz_j \cdot \delta]$,其中 wz_j 为格 c 在第 j 维上从原点开始的位置,δ 为格的宽度.另一方面,已经一个对象,我们也很容易计算出它落入哪一个格内.例如对于 k 维对象 $p(i.attr_1, \cdots, p.attr_d)$,包含 p 的格 $c(wz_1, \cdots, wz_d)$的各维位置为:$wz_j = \lceil p.attr_j/\delta \rceil, j \in [1, k]$.此外,每个格关联一个指针列表,用来保存指向各对象的指针.

在局部查询计算阶段,我们假设超点计算机 SP 与 m 台简单计算机相关联,那么 SP 上总共有 $m+1$ 个 S_f 集合,我们标记为 $S_f^{(0)}, S_f^{(1)}, \cdots, S_f^{(m)}$,其中 $S_f^{(0)}$ 为 SP 本地的 S_f 集合,而 $S_f^{(1)}, \cdots, S_f^{(m)}$ 为与 SP 相关联的 m 台简单计算机传输过来的 S_f 集合.对于 k 维对象 p,我们用 $k+2$ 个属性来表示,即 $p = (attr_1, \cdots, attr_k, flag, pos)$,其中前 k 个属性对应它的 k 个维度,而第 $k+1$ 个属性标识对象 p 所在的 S_f 集合,第 $k+2$ 个属性记录对象 p 在 S_f 中的位置.在全局查询计算阶段,我们假设分布式网络中存在 $n+1$ 台超点计算机,那么 Initiator 上总共有 $n+1$ 个局部子空间轮廓集合,我们标记为 $sky(V)^{(0)}$,

$\text{sky}(V)^{(1)},\cdots,\text{sky}(V)^{(n)}$，其中 $\text{sky}(V)^{(0)}$ 为 Initiator 本地的 $\text{sky}(V)$ 集合，而 $\text{sky}(V)^{(1)},\cdots,\text{sky}(V)^{(n)}$ 为其余 n 台超点计算机传输过来的子空间轮廓集合. 由于在全局查询计算阶段，我们不需要标识对象的来源以及它所在的位置信息，因此，当 Initiator 处于全局查询计算阶段时，对象只用 k 个属性来表示，即 $p=(\text{attr}_1,\cdots,\text{attr}_d)$.

由于在局部查询计算阶段，我们是从全空间轮廓集合 S_f 中获取子空间上的种子轮廓对象；而在全局查询计算阶段，我们处理的是子空间轮廓集合；因此，在给出 ASSC 方法之前，我们先给出一个定义，即格投影.

定义 1（V-投影格） 假定全空间 F 由 k 个维组成，即 $F=\{d_1,\cdots,d_k\}$，而子空间 V 由 $v(v\leqslant k)$ 个维组成，不失一般性，假设 V 由前 v 个维组成，即 $V=\{d_1,\cdots,d_v\}$；如果子空间 V 上的格 c^V 满足如下条件，那么我们称 c^V 为全空间 F 上的格 c^F 的 V-投影格：$\forall i\in[1,v]$，$\text{wz}_i(c^V)=\text{wz}_i(c^F)$，$\text{wz}(c^V)$ 为格 c^V 在第 i 维的位置.

ASSC 方法的执行过程如下：①ASSC 方法首先获取和组织需要处理的所有对象集合 S_{req}. 对于局部查询计算阶段，$S_{\text{req}}=\bigcup_{i=0}^{m}S_f^{(i)}$，而对于全局查询计算阶段，$S_{\text{req}}=\bigcup_{i=0}^{n}\text{sky}(V)^{(i)}$. 并且将 S_{req} 划分为 h 个组 $G=\{g_1,\cdots,g_h\}$，其中每一组的所有对象均落入同一个 V-投影格中. 不难看出，每一个组 $g_i(1\leqslant i\leqslant h)$ 对应一个不同的 V-投影格 c_i，即 c_i 包含组 g_i 中的所有全空间轮廓对象. ②之后，ASSC 方法对这 h 个格按排序算子 $\prod_{t=1}^{v}\text{wz}_t(c_i)$ 非递减排序，其中 $\text{wz}_t(c_i)$ 为格 c_i 的第 t 维的位置，记排序后的格序列 OS 为 $\text{ec}_1,\cdots,\text{ec}_h$. ③接着，从 ec_1 开始，ASSC 方法依次将序列 OS 中的格 $\text{ec}_j(1\leqslant j\leqslant h)$ 与它前面的格进行比较，如果在它前面的格中，存在一个格 e，使得 e 在所有 v 个维上的位置值均比 ec_j 小，我们称格 ec_j 为不活动格，那么从序列 OS 中删除格 ec_j，因为格 ec_j 上的所有对象不可能成为 V 上的种子轮廓对象. 如果 ec_j 是活动格，那么首先过滤掉格 ec_j 中在子空间 V 上被同一个格中的其它对象支配的所有对象，记保留下来的对象集合为 S_j. 然后，对于 S_j 中的每个对象 p，将 p 与在 ec_j 之前且满足如下条件的格 χ 中保留下来的所有对象进行比较：格 χ 在任何一个维上的位置均不比 ec_j 大，并且至少在一个维上的位置比 ec_j 小. 如果 p 在子空间 V 上被这些对象支配，则过滤掉 p. 当处理完 OS 中的最后一个格时，格序列里所有活动格中保留下来的对象的并集即为所求的结果 S_{result}. ④最后，如果是局部查询计算阶段，我们必须对 S_{result} 中的对象按标识属性 flag 进行分类，并获取每个对象的位置值 pos，同时，将每一类中的对象的位置值进行非递减排序并回送给相应的简单计算机；如果是全局查询计算阶段，我们只需简单地返回 S_{result} 给用户即可.

基于上面的分析，ASSC 方法的伪码可描述如下.

算法 2 ASSC 方算法

/* 如果是局部查询计算阶段，回送非递减排序的种子轮廓对象的位置信息给相应的简单计算机；如果是全局查询计算阶段，返回最后结果给用户 */

```
Begin
1.  if 轮廓查询处于局部查询计算阶段 then
2.    for i = 1 to m do
3.      接收第 i 台简单计算机传送的 S_f^(i);
4.    S_req ← Y_{i=0}^{m} S_f^(i);
5.  if 轮廓查询处于全局查询计算阶段 then
6.    for i = 1 to n do
7.      接收第 i 台超点计算机传送的 sky(V)^(i);
8.    S_req ← Y_{i=0}^{n} sky(V)^(i);
9.  S_result ← ∅;
10.   将 S_f 划分为 h 个组，每一个组 g_i(1≤i≤h) 对应一个不同的 V-投影格 c_i;
11.   将这 h 个格组织为序列 OS，满足：∀ i,j∈[1,h], i < j ⇒ ∃ t∈[1,v], wz_t(c_i) < wz_t(c_j);
12.   for 顺序访问 OS 中的每个格 c_i do
13.     if 在它前面的格中，存在一个格 e，满足：
        ∀ j∈[1,v], wz_j(e) < wz_j(c_i) then
14.       从 OS 中删除格 c_i;
15.     else
16.       过滤掉 c_i 中在 V 上被 c_i 中的其它对象支配的所有对象，记保留下来的对象集合为 S_i;
17.       for OS 中在 c_i 之前的每个格 e do
18.         if ¬ ∃ x∈[1,v], wz_x(e) > wz_x(c_i) 且
                ∃ t∈[1,v], wz_t(e) < wz_t(c_i) then
19.           for c_i 中的每个格对象 p do
20.             if p 被 χ 中对象支配 then
21.               从 S_i 中过滤掉对象 p;
22.       S_result ← S_result ∪ S_i;
23.   if 轮廓查询处于局部查询计算阶段 then
24.     for q = 1 to m do
25.       plist^(q) ← ∅;
          /* plist^(q) 回送给第 q 台简单计算机的位置值列表 */
26.     for S_result 中的每个对象 r do
27.       plist^(r.flag) ← plist^(r.flag) ∪ {r.pos};
28.     for i = 1 to m do
29.       按 pos 的值非递减排序 plist^(i);
30.       回送 plist^(i) 给简单计算机 SC_i;
31.   if 轮廓查询处于全局查询计算阶段 then
32.     返回 S_result 给用户;
End
```

定理 2（ASSC 方法的正确性） 当 ASSC 方法终止时，S_{result} 集合由 $\text{SET}_{\text{input}}$ 在子空间 V 上的轮廓对象构成.

4 优化策略

在这一节中，我们进一步优化子空间 V 上的轮廓

查询的效率.当使用逐层传送和处理机制时,我们给出一种能够进一步降低计算时间和数据传输量的优化策略,即最小格位置传送策略.第 g 层超点计算机的最小格位置传送策略的有效程度取决于第 $g+1$ 层各超点计算机间查询计算时间开销的差异程度.接下来,我们给出最小格位置传送策略的定义.假定 SPA 结构的分布式网络具有 z 个层.第 $g(0<g\leqslant z)$ 层的超点计算机 SP_g 传送给上一层超点计算机的数据格式为 $\{lev_{g\_}sky(V),<wz_1(minC),\cdots,wz_v(minC)>\}$,其中 $lev_{g\_}sky(V)$ 为 SP_g 上的输出对象集合,minC 为 SP_g 执行 ASSC 算法后的格序列 OS 中的第 1 个格,v 为子空间 V 的维数,而 $wz_i(minC)$ 表示格 minC 的第 i 维的位置值.因为各超点计算机上的数据对象个数不同,所以,它们执行 ASSC 算法的时间存在差异,从而使得上一层超点计算机不会同时接收到与它关联的下一层超点计算机传送来的数据.

定义 2(最小格位置传送策略)　假定与第 $g-1$ 层超点计算机 SP_{g-1} 相关联的第 g 层超点计算机有 l 台 $SP_g^{(1)},\cdots,SP_g^{(l)}$.不失一般性,假定 l 台第 g 层超点计算机按照这个顺序完成 ASSC 算法.SP_{g-1} 循环完成如下工作,直到接收完所有数据集为止:在第 u 次循环时,SP_{g-1} 接收到 $t(0<t<l)$ 台第 g 层超点计算机传送给它的数据集之后,找出 t 个格中的最小格 $mC_m^{(u)}$,$mC_m^{(u)}$ 满足如下性质:$thd(mC_m^{(u)})=\sum_{w=1}^{v}wz_w(mC_m^{(u)})$ 的取值在所有 t 个格中最小.并且将 $thd(mC_m^{(u)})$ 与先前循环获取的 $u-1$ 个最小格比较,若 $thd(mC_m^{(u)})$ 值比先前的 $u-1$ 个最小格都小,那么 SP_{g-1} 将 $<wz_1(mC_m^{(u)}),\cdots,wz_v(mC_m^{(u)})>$ 传送给其余 $l-u\cdot t$ 台未传送数据给 SP_{g-1} 的第 g 层超点计算机,在这些计算机上执行 ASSC 算法的过程中,将 $mC_m^{(u)}$ 插入于格序列 OS 的第 1 个格之前,否则不传送任何格信息,只是等待接收其余 $l-u\cdot t$ 台超点计算机传送过来的数据.

根据定义 2,我们可以得出最小格位置传送策略降低计算时间和数据传输量的原理:如果在超点计算机 SP_1 上的格序列 OS_1 中,格 c 无法通过 OS_1 的其它格来推导出它为不活动格,那么 ASSC 方法需要进行关于 c 的格内对象比较和格间对象比较.而最小格位置传送策略显著增加了 c 为不活动格的概率,从而降低了 ASSC 方法进行对象间比较的次数.因此,最小格位置传送策略能够显著降低了计算时间.另一方面,由于格 c 中本来需要传送的那些对象,当格 c 被判断为不活动格之后,这些对象就无需再进行传送,因此,最小格位置传送策略也降低了数据传输量.

5　实验评估

这一节通过具体的实验来评估本文方法和技术的有效性.实验评估环境为:PIV 2.0GHz CPU、2GB 内存、120GB 硬盘;Windows XP 操作系统.所有实验代码在 java 编译器中运行通过.

我们使用文献[2]中的反相关分布数据集.为了简单而不失一般性,数据集共有 8 个属性.对象所有维属性、对象位置值以及重复值计数器均为整型,长度为 Bytes.Bloom Filter 中使用的两个哈希函数为:

$$h_1(key)=Mod(\lfloor key\rfloor,LZ),$$

$$h_2(key)=\lfloor LZ\cdot Mod(0.618\cdot key,1\rfloor),$$

其中 LZ 为全空间扩展轮廓集合与全空间轮廓集合之间差集的对象个数.多维正规格结构将每个维度划分成 5 个子区域.此外,不失一般性,我们假定维属性取值越小,那么在该维上的优越程度越高.

在实验中,我们仿真具有 1111 个节点的 SPA 架构分布式网络.该仿真网络具有 4 个分布式层次.其中一个为接收用户查询的超点计算机 Initiator,它连接 10 个一级超点计算机,每个一级超点计算机连接 10 个二级超点计算机,而每个二级超点计算负责 10 个简单计算机.与文献[2]一样,网络带宽取 40k bytes/s.此外,用户在 Initiator 超点计算机上发出 60 个子空间轮廓查询.在实验中,我们分四种情况来组织这 60 个子空间轮廓查询,分别为:

(1)QS1:60 个查询的子空间全部是 4 个维度;

(2)QS2:6 维子空间 15 个、5 维子空间 18 个、4 维子空间 23 个和 3 维子空间 4 个;

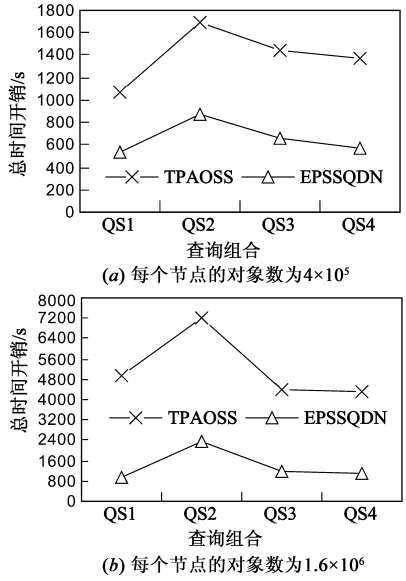
(3)QS3:6 维子空间 4 个、5 维子空间 13 个、4 维子空间 16 个、3 维子空间 17 个和 2 维子空间 10 个;

(4)QS4:7 维子空间 1 个、6 维子空间 5 个、5 维子空间 11 个、4 维子空间 21 个、3 维子空间 9 个和 2 维子空间 13 个.

在上面的实验环境下,比较本文 EPSSQDN 算法和文献[2]中所给出的 TPAOSS 算法总时间开销.图 1 给出了实验结果.

6　结论

传统 C/S 架构的网络能够方便地升级到 SPA 体系架构的分布式网络,因此研究"在 SPA 架构的分布式网络中有效进行子空间轮廓查询"是一个很有意义的工作.本文分析了文献[2]在解决 SPA 网络中子空间轮廓查询方面存在的两个个主要性能缺陷,并给出一种在 SPA 网络中,进行子空间轮廓查询的有效方法 EPSSQDN.基于 EPSSQDN 算法,我们提出有效的优化策略来缩短 SPA 网络中子空间轮廓查询的总时间开销.



(*a*) 每个节点的对象数为4×10^5

(*b*) 每个节点的对象数为1.6×10^6

图1 算法性能对比实验

此外,我们从理论上分析和证明了本文算法的正确性以及有效性.同时,本文实施大量的实验评估,并从多个不同的角度来综合考察本文算法的实用性和可扩展性.

本文将来的后续工作主要包括:提出新的数据结构来替代扩展轮廓集合,从而进一步提高获取重复对象的效率以及降低 Bloom Filter 的错误定位概率;研究不同架构的分布式网络中的子空间轮廓查询处理.

参考文献

[1] S Borzsonyi, D Kossmann, K Stocker. The skyline operator [A]. Proc IEEE ICDE '01 [C]. Heidelberg: IEEE Press, 2001.421 – 430.

[2] 黄震华,王智慧,郭建奎,汪卫,施伯乐.有效预处理 P2P 网络中的子空间 skyline 查询[J].软件学报,2009,20(7):1825 – 1838.
Z Huang, Z Wang, J Guo, W Wang, B Shi. Efficient preprocessing of subspace skyline queries in P2P networks[J]. Journal of Software, 2009, 20(7): 1825 – 1838. (in Chinese)

[3] P Wu, C Zhang, Y Feng, B Zhao, D Agrawal, A Abbadi. Parallelizing skyline queries for scalable distribution [A]. Proc EDBT'06[C]. Munich: Springer Verlag, 2006.112 – 130.

[4] H Li, Q Tan, W Lee. Efficient progressive processing of skyline queries in peer-to-peer systems [A]. Proc INFOSCALE' 06 [C]. Hong Kong: ACM Press, 2006.84 – 93.

[5] S Wang, B Ooi, A Tung, L Xu. Efficient skyline query processing on peer-to-peer networks[A]. Proc ICDE'07[C]. Istanbul: IEEE Press, 2007.372 – 381.

[6] K Banafaa, R Li. Efficient algorithms for constrained subspace skyline query in structured peer-to-peer systems [A]. Proc WAIM '12[C]. Harbin: Springer Verlag, 2012.334 – 345.

[7] 薛小平,张思东,张宏科,王小平,葛乐,尹琴.基于内容的发布订阅系统路由算法[J].电子学报,2008,36(5):953 – 961.
X Xue, S Zhang, H Zhang, X Wang, L Ge, Q Yin. Content-based routing algorithms of the publish-subscribe systems[J]. Acta Electronica Sinica, 2008, 36(5): 953 – 961. (in Chinese)

[8] J Parreira, S Michel, G Weikum. P2P Dating: Real life inspired semantic overlay networks for web search[J]. Information Processing and Management: An International Journal, 2007, 43 (3): 643 – 664.

[9] K Zhao, Y Tao, S Zhou. Efficient top-k processing in large-scaled distributed environments[J]. Data & Knowledge Engineering, 2007, 63(2): 315 – 335.

[10] 吴万明,吴毅坚,赵文耘.基于 Chord 网的语义 Web Service 发现 [J].电子学报,2007,35(z2):152 – 155.
W Wu, Y Wu, W Zhao. Chord-based semantic web service discovery[J]. Acta Electronica Sinica, 2007, 35(z2): 152 – 155. (in Chinese)

[11] X Liu, Y Yuan, W Wang, H Lu. Stabbing the sky: efficient skyline computation over sliding windows[A]. Proc IEEE ICDE'05 [C]. Tokyo: IEEE Press, 2005.502 – 513.

[12] Q Li, L Lopez, B Moon. Skyline index for time series data[J]. IEEE Transactions on Knowledge and Data Engineering, 2004, 16(6): 669 – 684.

作者简介

黄震华 男,1980 年生,博士,副教授,软件行业协会系统工程分会理事,CCF 会员.主要研究方向为云计算、信息服务、数据挖掘与知识发现等.
E-mail: jukie.huang@gmail.com

向 阳 男,1962 年生,教授,博士生导师.主要研究方向为智能计算、数据仓库、数据挖掘、决策支持系统与语义网等.

孙圣力 男,1979 年生,博士,讲师,CCF 会员.主要研究方向为数据库、数据流挖掘与服务计算等.

陈 千 男,1984 年生,博士研究生.主要研究方向为数据挖掘、模式识别和服务发现等.