云环境下基于超球面投影分区的Skyline计算

雷 婷[1] 王 涛[2] 曲 武[3,4,5] 韩晓光[6]

(成都工业学院通信工程系 成都 611730)[1] (湖南城市学院信息科学与工程学院 益阳 413000)[2]

(清华大学知识工程研究室 北京 100083)[3] (北京启明星辰信息技术股份有限公司 北京 100193)[4]

(中关村科技园区海淀园企业博士后科研工作站 北京 100193)[5]

(北京科技大学计算机与通信工程学院 北京 100083)[6]

摘 要 目前,Skyline查询在集中式数据库、分布式数据库、数据流及分类属性数据集上的良好应用前景,使其成为当前数据库界研究的重点和热点之一,受到了学术界和工业界的广泛关注,它作为一种重要的数据挖掘技术广泛应用于多目标优化、城市导航系统、用户偏好查询及约束决策、智能防御系统以及地理信息系统等领域。随着人类可以采集和利用的数据信息的急剧增长,如何处理大数据的Skyline查询成为急需解决的问题。针对云计算环境,在Map-Reduce框架下设计并实现了基于超球面投影分区的分布式Skyline算法HSPD-Skyline,其主要思想是通过对高维数据点的超平面投影映射,即由空间坐标转换为超球面坐标,可以有效提高分区内数据点的平均减枝力度,降低Skyline的计算代价。同时,使用基于空间分区树的启发式策略HA-SPT,进一步提高了HSPD-Skyline算法的处理效率。通过详细的理论分析和实验验证表明,在不考虑数据分布和进一步优化算法的条件下,提出的HSPD-Skyline算法的总体性能(可扩展性、Skyline查询时间等)优于同类算法。

关键词 分布式Skyline计算,Map-Reduce框架,分区策略,HSPD-Skyline算法

中图法分类号 TP391 **文献标识码** A

Distributed Skyline Processing Based on Hypersphere Projection Partitioning on Cloud Environments

LEI Ting[1] WANG Tao[2] QU Wu[3,4,5] HAN Xiao-guang[6]

(Communication Engineering, Chengdu Technological University, Chengdu 611730, China)[1]

(Hunan City College of Information Science and Engineering, Yiyang 413000, China)[2]

(Tsinghua University Knowledge Engineering Group, Beijing 100083, China)[3]

(Core Research Institute, Beijing Venustech Cybervision Co. Ltd., Beijing 100193, China)[4]

(Enterprise Postdoctoral in the Zhongguancun Haidian Science Park, Beijing 100193, China)[5]

(School of Computer & Communication Engineering, University of Science & Technology Beijing, Beijing 100083, China)[6]

Abstract Recently, skyline processing has been receiving considerable attention due to its potential applications in many fields, including traditional database, distributed database, data stream and even the categorical database and so on. Both the academic and the industrial have paid much attention to it. As an important data mining technique, skyline processing is of great significance for multi-objective optimization, urban navigation, multi-criteria decision making and preference query, trip planning, defense and intelligence systems and geographic information systems. In addition, the amount of data collected and used by human is developing at an astonishing speed. Therefore, how to process Skyline query of massive data is an urgent problem. Aiming at cloud computing applications, this paper designed and implemented distributed Skyline processing based on hypersphere projection partitioning under the Map-Reduce framework, HSPD-Skyline. It is showed that partitioning the data according to the hyperspherical coordinates can increase the average pruning power of points within a partition, and reduce the cost of Skyline processing. The HSPD-Skyline algorithm also uses a heuristic strategy based on space partitioning tree, HA-SPT, to further improve the processing efficiency of the HSPD-Skyline algorithm. Finally, the theoretical analysis and experiment results illustrate that the HSPD-Skyline algorithm (Distributed Skyline Processing based on Hypersphere Projection Partitioning) consistently outperforms similar approaches for distributed skyline computation, regardless of data distribution, and further optimization strategies.

Keywords Distributed Skyline processing, Map-Reduce frame, Partitioning strategy, HSPD-Skyline

1 引言

许多现实应用中,例如社交网站好友推荐、电子商务商品推荐、股票交易优股推荐、购买房产推荐等等,用户往往希望从大的数据集中得到"最优"的选项,这是典型的多目标优化问题。目前,存在很多解决此类问题的算法,根据采用的手段

到稿日期:2012-08-20 返修日期:2012-12-25 本文受基于大规模复杂结构知识库的知识发现机理、模型与算法研究(60875029),多关系频繁模式挖掘模型、方法与一般架构的研究(60675030),基于多关系的模糊认知图挖掘模型、算法与评价机制研究(61175048)资助。

雷 婷(1984—),女,硕士,讲师,主要研究方向为海量数据挖掘、云计算,E-mail:tinglei.uestc@gmail.com(通信作者);王 涛(1981—),男,硕士,主要研究方向为数据挖掘;曲 武(1981—),博士生,主要研究方向为数据挖掘、云计算、社交网络分析;韩晓光(1981—),男,博士,主要研究方向为云计算、网络安全。

不同，主要分为3类：

1)Top-K查询，转化成单目标优化问题。

2)Skyline查询，不是将问题转化为单目标优化问题，而是直接采用多目标优化算法解决原始的多目标问题。多目标优化算法最终返回的结果集是一系列平行的、互不受控制的解，即多个Skyline点。这里的“控制”是指一个数据点在每一维上都不比另一点“差”，而且必须至少在一维上比另一点要“好”。其中，“差”和“好”并无统一的定义，根据用户的查询和选择条件的语义而定。以用户购买房产为例(见图1)，数据集中每个元组都是一个数据点，X维表示房产到市中心的距离，Y维表示房产价格，通过Skyline的定义，距离市中心近的且价格低的房产控制着其他房产，是用户希望买到的。因此，Skyline点是价格和距离最有可能的折衷。

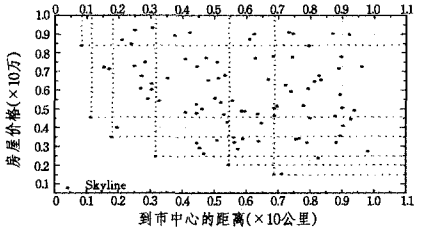

图1 Skyline实例

3)词典序方法，是数据库中最常使用的方法，即为不同的属性安排不同的优先级，然后按照优先级的高低来选择各个属性进行优化。

事实上，Skyline计算本身不是一个新的课题。Skyline计算工作可以追溯到计算向量集中最大向量(Maximal vector)问题[1]。最大向量问题是要在向量集中找到不被别的向量支配的所有向量的集合。所谓一个向量支配另一个向量，是指该向量的每一个要素都不比另一个向量对应要素小，且至少存在一个要素大于另一向量对应要素。同样地，一个向量也可视作高维空间中的一个点，在这样的语义下，最大向量也称为“可取点”(admissible point)，所有可取点的集合叫做Pareto集。该问题从提出到现在已经被不少学者研究了很多年，提出很多算法，这些算法关注不同限制条件下最大向量问题的计算复杂度，不考虑处理的数据超出内存的情况，因此是典型的基于内存的算法。数据库领域的研究人员对Skyline查询的研究始于2001年，最早由Borzsonyi等人提出[2]，主要关注在数据量很大、无法放入内存的情况下，如何处理Skyline查询。近几年，在SIGMOD、VLDB、KDD、PODS、ICDE、ICDM等相关的高水平国际会议上有许多高质量的Skyline研究论文发表，展示出大量的研究成果。在TKDE，TODS等期刊中也有大量成果发表。最近几年，对Skyline查询的研究大体上可以分为4类：(1)集中式数据库Skyline查询；(2)数据流上Skyline查询；(3)分布式环境下Skyline查询；(3)P2P环境下Skyline查询。鉴于分布式Skyline查询处理在多目标优化应用方面的重要价值和在实时在线服务方面的良好应用前景，本文将着重研究分布式Skyline查询。

本文在云环境下提出一种基于超球面投影分区的Skyline计算(HSPD-Skyline，Distributed Skyline Processing based on Hypersphere Projection Partitioning)。该算法通过将空间坐标投影到超球面上转化为超球面投影坐标，然后使用超球面投影坐标进行分区，可以有效地提高分区内数据点的平均减枝力度，降低Skyline的计算代价。通过详细的理论分析和实验验证表明，在不考虑数据分布和进一步优化算法的条件下，本文提出的HSPD-Skyline算法的总体性能(可扩展性、Skyline查询时间等)优于同类算法。

本文第2节介绍了相关概念及研究背景；第3节提出了云环境下基于超球面投影分区的Skyline算法HSPD-Skyline，并详细介绍了该算法的基本思路和实现策略；第4节对HSPD-Skyline方法与随机投影分区算法、格分区算法进行实验比较和分析；最后进行小结，展示未来的研究方向。

2 相关概念及研究背景

2.1 相关概念

将数据空间D定义为一个d维$\{d_1,\cdots,d_d\}$数据点(对象)的集合。一个D上的数据集P其规模为n，数据点$p\in P$能够被表示为$p=\{v_1,\cdots,v_d\}$，v_i为d_i维的值，定义d_i维的值范围是$0\leqslant v_i\leqslant 1$。

定义1(支配) 任意给定两个数据点$p_i,p_j\in P$。若数据点p_i在任意一个维度上的取值都不小于数据点p_j，且至少在某一维度上的取值p_i比p_j要大，则定义p_i在数据空间D上支配p_j，记为$p_i<p_j$。若p_i在D的某个子空间(子分区)Su上支配p_j，则记为$p_i \underset{S}{<} p_j$。

定义2(Skyline) d维数据空间D上，P中所有不被其他数据点支配的数据点组成的集合，记为S。

定义3(子空间(子分区)Skyline) 在数据集P的某一个子空间$P_i(P_i\subseteq P,P_i\neq\emptyset)$上，$P_i$中所有不被其他数据点所支配的数据点组成的集合，记为S_{P_i}。

假设，使用S_i个Slave计算节点进行分布式Skyline计算，使用空间分区方法将数据集P水平切分为N个子分区，并存储到分布式文件系统HDFS中，对于所有$i\neq j$，有$P_i\subseteq P$，$\mathrm{U}_{1\leqslant i\leqslant N}P_i=P$，$P_i\cap P_j=\emptyset$。得出，若数据点$p\in P$，有$p\in S_{P_1}$，当且仅当存在一个分区$P_i(1\leqslant i\leqslant N)$，$p\in P_i\subseteq P$，并且$p\in S_{P_1}$。即，第一阶段，对于水平分区的数据集，全局Skyline点是所有分区中局部Skyline点集合的子集。每个Slave计算节点S_i处理数据集分区P_i，计算局部Skyline，S_{P_1}。第二阶段，通过计算局部Skyline集合的Skyline获取全局Skyline。以上的说明确保了分布式Skyline算法能够获得精确的全局Skyline。表1给出了常用的符号说明。

表1 文中所用符号说明

符号	描述
D	数据空间，一个d维$\{d_1,\cdots,d_d\}$数据点(对象)的集合；
d	数据维度；
P	数据空间D上的一个数据集P；
n	数据集规模；
p	数据点$p\in P$能够被表示为$p=\{v_1,\cdots,v_d\}$，p_i为d_i维的值，定义d_i维的值范围是$0\leqslant p_i\leqslant 1$；
p_i	p_i为数据点p的d_i维值，定义p_i值范围是$0\leqslant p_i\leqslant 1$；
θ	数据点$p=\{p_1,\cdots,p_d\}$，被映射到超球面上的球面坐标；
θ_i	θ_i为球面坐标θ的第i维的值；
r	超球面坐标的半径坐标；
P_i	第i个分区的数据集；
R	分区数目；
S_{P_i}	子分区P_i中所有不被其他数据点所支配的数据点组成的集合

2.2 研究背景

目前，Skyline计算因在集中式数据库、分布式数据库、数

据流及分类属性数据集上的良好应用前景，而成为当前数据库界研究的重点和热点之一，受到了学术界和工业界的广泛关注。Börzsönyi 等人首先在数据库领域探索了 Skyline 计算问题，提出了 BNL 算法[2]，基本原理是通过将数据集中某个数据点与数据集中其它数据点进行比较来检索数据集中不被其它数据点支配的数据点。D&C(divide-and-conquer)算法[2]将数据集划分为几部分，使得每部分都可以放入内存，然后使用内存算法分别计算每一部分的 Skyline 集合，最后通过合并每部分的 Skyline 集合去除其中受控的数据点来得到最终的 Skyline 集合。SFS(sort-filter-skyline)算法[3]在 BNL 算法的基础上对数据集进行了预排序。BNL 算法的 I/O 开销依赖于迭代的次数和每次迭代要处理的数据集的大小。因为内存中窗口大小有限，不一定能装下所有的 Skyline 集合，所以对要处理的元组先排序，过滤掉一些受控元组，因此出现了 SFS 算法。SFS 算法的基本思想是对临时表中的元组按照 Skyline 集合的选取规则先做一个拓扑排序，然后用 BNL 算法处理这些元组。故一旦有一个临时表 Ti 中的元组被加入窗口中，它就一定是 Skyline 集合了。因为 Ti 有序，所以它后面的元组不可能控制它，这样就不需要做替换的检查，减少了开销。Bitmap 算法[4]是采用位图结构来判断一个点是否是 Skyline 集合，算法满足渐进性，无须遍历整个数据集，找到一个 Skyline 集合便可以及时返回，因为按位操作计算速度很快，故可以很快地判断出一个点是否是 Skyline 集合。LESS (linear elimination sort for skyline)算法[5]结合了 SFS，BNL 和 FLET 算法的某些方面，是 SFS 算法的改进算法。NN 算法[6]的基本原理是先对数据集中的数据点构建 R-Tree 索引结构，并利用最近邻算法求出距离原点最近的数据点，然后利用该数据点的信息对数据空间进行划分，并在每个划分中递归此过程，直到分区中不包含任何 Skyline 点位置。文献[7]提出一种在线高效子空间 Skyline 算法 CSky，其利用新的数据结构 InventS 将可能成为 Skyline 结果的点置于优先扫描的位置，保证了算法的渐进性。文献[8]为解决时序数据上 Skyline 查询问题，提出了基于小波概要的区间差分 Skyline 查询算法，以支持不同粒度区间差分 Skyline 查询。文献[9]为解决道路网络环境下移动对象的连续概率 Skyline 查询问题，设计了对网络受限的不确定移动对象进行连续概率 Skyline 查询的动态增量算法 U-CPSQRN。该算法通过对事件的跟踪计算实现了对 p-Skyline 的连续更新操作，减少了算法的查找和计算开销。

为了更好地适应在不同环境下的应用，最近两年对 Skyline 问题的研究逐渐地趋向于在具体应用环境下进行，如分布式[10-14]和并行[15-18]环境。在这两种情况中，研究人员通过对数据源进行垂直分区和水平分区的方式，将数据集分布到 N 个计算节点上。为了处理分布式环境下的 Skyline 查询，文献[10]首次提出了一个有效的分布式 Skyline 查询算法；同时，还利用两条启发式规则提出了该算法的改进算法，提前及时地终止了每阶段的检测工作，从而提高了效率；并且为了避免“维度诅咒”的问题，作者提出了一种新型的抽样方案，以便尽快得到大致的 SP 集合，然后再细化改良。文献[19,20]使用多台机器来有效地处理分区数据关系上的查询。文献[16]提出一个不同的并行 Skyline 计算方法，作者使用单处理器多磁盘的架构，与串行相反，它们使用并行 R-树。为了提高对非 Skyline 点的剪枝效率，该方法并发访问多个磁盘的多个入口。文献[21]对计算几何领域的并行 Skyline 查询进行研究，提出了一个最佳的粗粒度 3 维并行 Skyline 算法，它采用分而治之的方法解决这个问题，但他们的方法在高维数据点集上的可扩展性较差。文献[22]结合分而治之的算法[23]和分支定界算法[24]，提出了新的并行 Skyline 算法。该算法首先以随机方式把数据集分配给参与计算的机器，以确保每个分区的数据组成类似于原始数据集。然后，每一台机器用 R-Tree 作为索引结构来对本地数据进行 Skyline 查询。

文献[25]提出了 MANET 上的第一个 Skyline 查询算法(以下简称为 MANET 算法)，它将所有移动设备组织成一个树状结构，并以发出查询的设备作为树的根节点。Skyline 查询从根节点逐层传递下去，每个节点收集由它所有子节点传来的中间 Skyline 集，与自己本地 Skyline 归并求出新的中间 Skyline 集并向上传递给父节点。这个过程被递归地调用直到所有中间 Skyline 集在根节点归并，得到最终的 Skyline 结果。P2P 网络中的数据分布存储在不同的节点上，每个节点既是客户机，又是服务器，没有统一的中央控制节点。P2P 上的 Skyline 计算算法不仅仅需要满足集中式 Skyline 计算算法的各种要求，还需要考虑减小网络通讯量，减少平均节点访问数，保持负载平衡等重要衡量标准。文献[15]提出了对等网络上的第一个 Skyline 计算算法，它是基于对等网络的一种重要拓扑结构，即 CAN 网络。网络中每个节点对应数据空间中的一个矩形区域，并存储所有位于这个矩形区域中的数据点。数据区域之间有着一定的支配依赖性，算法通过动态地为区域编号标识出区域间的依赖关系，并按此依赖关系访问各个相关节点，并行地传递和执行 Skyline 查询，逐步得出全部的 Skyline。文献[26]研究了 P2P 网络环境下的 Skyline 查询处理问题，提出了一个适用于 P2P 网络环境下的 Skyline 查询处理算法。对于更广泛的 P2P 网络，尽管节点可以分簇，但是并不能使用语义信息来划分和管理簇。文献[13]为有超级节点的 P2P 网络中的子空间的 Skyline 查询提出了一个基于阈值的算法，即 SKYPEER。为了有效地处理子空间上的 Skyline 查询，Wang 等人[26]提出了 Skyline 空间分区方法 SSP，其使用一个树型结构的 P2P 平台 BATON 来计算 Skyline。SSP 把 Skyline 空间分布到区域，使用 Z-曲线方法将多维数据空间映射到一维 BATON 上。在此基础上，文献[14]提出了基于树型对等网络 BATON 的 Skyline 计算算法，它将整个数据空间适应性地划分为对应于网络节点的各个区域，Skyline 查询被按照区域依赖关系发送到可能包含 Skyline 点的节点上，以减少访问的节点数和传递的信息量。

并行 Skyline 计算也在文献[27]中进行了研究。作者为每个数据点随机选择一个计算节点。将数据点随机分区到计算节点中，便能以较大的概率保持每个分区的数据分布与原始数据集数据分布一致，因此随机分区方法能够满足公平要求。接下来使用 R-Tree 索引局部分区数据集。本文探索了随机分区的性能，并且与本文提出的方法进行了比较。

目前，并行或分布式 Skyline 查询处理大多使用空间划分方法，依赖于随机分区[27]和格分区[28,29]。随机分区方法容易实现，不添加任何的计算开销来决定每个点落入哪个分区。在计算过程中，每个分区独立处理，不会出现空或稀疏分区，可以保证分布式系统负载平衡。尽管随机分区满足大多数的

需求，但这种方法不能保证空间相近的数据点分到一个分区，即并没有通过分区步骤减少后期的处理代价。格分区是递归地将空间的维度划分为两个部分，维度为 d 的空间被分成 2^d 个分区，然后数据点根据分区边界落入各个分区。这个方法的每个计算节点负责处理一个分区，计算节点的连接方式反映了它们之间的通信方式。通过部分序列化计算节点之间的通信，确保返回正确的结果，同时避免连接所有计算节点，特别是对结果没有贡献的服务器。然而，这种方式也约束了检索执行的并行化，并且导致集群计算节点负载不平衡。

2.3 Map-Reduce 框架

Map-Reduce[30]是一个编程模式，它与处理或产生大数据集的实现相关。用户指定一个 Map 函数，通过这个 Map 函数处理 key/value（键/值）对，产生一系列的中间 key/value 对，并且使用 Reduce 函数来合并所有的具有相同 key 值的中间键值对中的值部分。使用这样的函数形式实现的程序可以自动分布到一个由普通机器组成的超大集群上并发执行。Run-time 系统会解决输入数据的分布细节，跨越机器集群的程序执行调度，处理机器的失效，并且管理机器之间的通讯请求。这样的模式允许程序员可以不需要有什么并发处理或者分布式系统的经验，就可以处理超大的分布式系统的资源。Map-Reduce 系统的实现运行在一个由普通机器组成的大型集群上，并且有着很高的扩展性：一个典型的 MapReduce 计算处理通常分布到上千台机器上来处理上 TB 的数据。使用者提供的 Map 和 Reduce 函数有着如下相关类型：

$$\begin{cases} Map(k_1, v_1) \rightarrow list(k_2, v_2) \\ Reduce(k_2, list(v_2)) \rightarrow list(v_2) \end{cases} \tag{1}$$

也就是说，输入的键值和输出的键值是属于不同的域。进一步讲，中间的键值和输出的键值是属于相同的域。

尽管 Map-Reduce 的相关研究已成为当今的研究热点，但是对于 Skyline 查询来说，利用 Map-Reduce 框架处理 Skyline 查询的相关研究才刚刚起步。目前，文献[31]将 Skyline 查询处理问题引入到 Map-Reduce 框架中，根据不同的数据划分策略，实现了 Skyline 查询的块嵌套循环算法（Block Nested Loops based on Map-Reduce，MR-BNL）等 3 个 Skyline 查询算法，文献[32]在此基础上提出了优化算法。但文献[31,32]并未从分区方法上对 Skyline 分布式算法进行优化，而本文针对大数据的 Skyline 查询中 Skyline 查询结果的数量远小于原始数据集的数据量的问题，结合文献[33]提到的角分区方法，提出在 Map-Reduce 框架下基于超球面投影分区的 Skyline 查询方法。实验结果表明，通过引入 Map-Reduce 框架，大大减少了计算超球面投影坐标的时间，避免了文献[33]的缺陷，并提出将分区时间作为总体 Skyline 计算时间的有效组成部分，通过与随机分区和格分区方法进行性能比较，HSPD-Skyline 算法总体性能优于二者。

3 HSPD-Skyline 算法的实现策略

本节主要说明 Map-Reduce 框架下的基于超球面投影的 Skyline 查询算法——HSPD-Skyline，如算法 1 所示。首先，3.1 节介绍近似 Skyline 方法，3.2 节介绍超球面坐标映射方法，3.3 节介绍基于超球面投影的分区方法，3.4 节介绍基于空间分区树的启发式策略。

算法 1 HSPD-Skyline 算法

输入：数据空间 D 上的一个数据集 P=points，$p_i \in P$，球面投影超球面投影坐标空间 A，$a_i \in A$，n=＃Slaves，d=dimension，任务参数 jobConf；

输出：数据集 P 的 Skyline 集合 S，表示为 GlobalSkyline_list；

```
BEGIN
//从 jobConf 中获取参数，初始化 Map-Reduce 任务；
1. InitMapReduce(jobConf,P);
2. //若数据集 P 的维度 d 为 1，直接跳到步骤 4，利用 Map-Reduce 机制直接计算 Skyline；
   If(d==1) then goto 4;
   else goto 3;
3. //Partition job，使用超球面投影对原数据集进行分区。其中 key1 为 point_id，value1 为数据空间 P 中的数据点；key2 为 point_id，value2为点 point_id 对应的超球面投影坐标。
Map(key1,value1,key2,value2)
{
//从 HDFS 中获取数据集；
  ReadFromHDFS(key1,value1);
//利用式(1)计算数据空间数据点 pi∈P 的超球面坐标；
  value2=Compute_Hyperspherepro(value1);
//输出超球面投影坐标；
  Output(key2,value2);
}
//Reduce 任务个数为 1，key3 为 partition_id，value2 为各个分区数据点集 point_list；
Reduce(key2,value2,key3,value3)
{
//根据格分区原理对于 d-1 维的超球面坐标进行分区，之后映射回原始坐标空间；
  key3=Compute_partition(value2);
//输出分区编号 key3 和分区内数据点列表 value3；
  Output(key3,value3)
}
4. //Skyline Job；
  Map(key3,value3,key4,value4)
  {
    ReadHDFS(key3,value3);
  //使用 SFS 算法计算各分区的 Skyline，也称为局部 Skyline，value4 为 Localskyline_list；
    value4=Compute_localSkyline(value3);
    Output(null,(key3,value4));
  }
  Reduce(key4,value4,key5,null)
  {
  //使用 SFS 算法计算局部 Skyline 集合的全局 Skyline，key5 为最终输出，GlobalSkyline_list；
    GlobalSkyline_list=Compute_globalSkyline(value4);
    Output(key5,null);
  }
END
```

3.1 近似 Skyline 方法

计算集合 P 的精确 Skyline 代价是巨大的，因为数据集中每个点都要和其他的点进行比较。然而，在 Skyline 集合较小的情况下，使用相对较少的比较消除大多数非 Skyline 点是

可行的。文献[34]对该方法进行了详细描述：首先，任意取一个小集合 B 作为比较点集，$B \subset P$；然后，比较数据集 P 中的每一个点是否被集合 B 中的点控制，那些被控制的点将被安全地丢弃。最后，近似的 Skyline 结果集合中确定包含所有的 Skyline 点和一部分非 Skyline 点。该方法的核心思想是较小的代价(一路扫描和线性时间)获取较大的减枝性能。使用近似 Skyline 方法不但可以减少传输和计算代价，而且能够减少出现负载不均衡的情况。这是通过预删除非 Skyline 点，确保各分区内 Skyline 点和非 Skyline 点比例均匀，从而使每个计算节点处理代价相近。HSPD-Skyline 使用 Map-Reduce 机制实现了近似 Skyline 方法，其作为 HSPD-Skyline 算法的数据处理阶段，在算法 1 中并没有给出。

3.2 超球面坐标映射方法

为使用超球面坐标进行分区，首先需要将数据点的笛卡尔坐标映射到超球面坐标，然后基于超球面坐标将数据空间分成 N 个分区。图 2 例证了超球面分区方法，左图是笛卡尔坐标空间中的分区，右图是经过超平面投影变换到超球面坐标的相同分区。笛卡尔坐标空间中的数据点 $p=\{p_1,p_2,\cdots,p_d\}$，映射到超球面坐标空间中，坐标包括半径坐标 r 和 $d-1$ 维的超球面投影坐标 $\theta_1,\theta_2,\cdots,\theta_{d-1}$，使用以下变换公式集：

$$\begin{cases} r=\sqrt{p_n^2+p_{n-1}^2+\cdots p_1^2} \\ \tan\theta_1=\dfrac{\sqrt{p_n^2+p_{n-1}^2+\cdots+p_2^2}}{p_1} \\ \cdots \\ \tan\theta_{d-2}=\dfrac{\sqrt{p_n^2+p_{n-1}^2}}{p_{n-2}} \\ \tan\theta_{d-1}=\dfrac{p_n}{p_{n-1}} \end{cases} \tag{2}$$

其中，一般情况下，对于 $i<d-1$，有 $0\leqslant\theta_i\leqslant\pi$，而对于 $i=d-1$，有 $0\leqslant\theta_{d-1}\leqslant2\pi$，本文实验验证算法中，对于 $i\leqslant d-1$，有 $0\leqslant\theta_i\leqslant\dfrac{\pi}{2}$。为了不失一般性，假设数据点 $p=\{p_1,p_2,\cdots,p_d\}$，对于 $\forall i$，有 $x_i\geqslant0$。

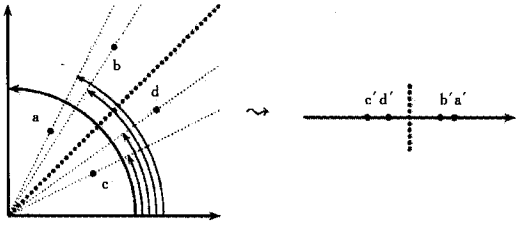

图 2 超球面投影分区方法示例图

3.3 基于超球面投影的分区方法

超球面投影分区首先映射笛卡尔坐标空间到超球面空间，然后使用超球面投影坐标 θ_i 将空间分成 N 个分区。实质上，是使用格分区算法[28,29]对 $d-1$ 维的超球面坐标空间进行分区。这将导致有空间相近的超球面坐标的数据点落入相同的分区，而与半径坐标无关，即不考虑数据点与原点之间的距离。对于分区数目 R 和一个 d 维的数据空间 D，角分区方法指定部分数据空间 $D_i(1\leqslant i\leqslant N)$ 给每个分区。数据空间第 i^{th} 分区定义：$D_i=\lceil\theta_1^{i-1},\theta_1^i\rceil\times\cdots\times\lceil\theta_{d-1}^{i-1},\theta_{d-1}^i\rceil$，其中，$\theta_j^0=0$，$\theta_j^N=\dfrac{\pi}{2}(1\leqslant j\leqslant d)$，第 i 个角分区 θ_j 的边界为 θ_j^{i-1} 和 θ_j^i。将部分数据空间 D_i 分配给每个分区之后，分布数据点到这些分区。首先，将每个数据点的笛卡尔坐标映射到超球面坐标，然后，将 $d-1$ 维的超球面投影坐标与分区边界进行比较，确定相应的分区，无需考虑到原点距离。角分区方法的核心思想是来自笛卡尔坐标同一方向的点可能位于不同的距离，这些点很可能被分配到同一个分区。通过这种方式，将与原点近的数据点分布到各个分区中，这可以避免原点附近密集的数据点落入一个或几个分区，控制一个特定的非 Skyline 点的点很有可能处于同一个分区，从而在局部 Skyline 计算时，允许这样的点被删除，以提高减枝能力。

3.4 基于空间分区树的启发式策略

本节将阐述 HSPD-Skyline 算法在合并局部 Skyline 过程中使用的基于空间分区树的启发式策略，即 HA-SPT(heuristic algorithm based on the space-partitioning tree)。分布式 Skyline 算法本质上是一个折中问题。在分配任务给不同的计算节点之前，数据预处理可以使计算局部 Skyline 过程和最终的合并过程更为高效。但另一方面，预处理时间也是全局运行时间的一部分。文献[33]提出的角分区方法，由于使用分区投影，复杂度较高，并没有设计相应的并行算法，因此其总体性能较低。寻找一个有效的平衡点，对于最小化全局时间是非常重要的。当合并局部 Skyline 数据集获得最终 Skyline 结果时，需要主要解决两个问题：(1)合并过程应当充分利用所有计算节点，即负载平衡；(2)通过减少局部 Skyline 点的比对计算次数，提高合并效率，降低合并时间。为解决问题(1)，本文提出使用空间分区树，如图 3 所示，执行一个自下而上的合并过程。分区树的每个节点存储当前的局部 Skyline，当需要删除非 Skyline 点时，再将各自的 Skyline 发送给其他节点进行操作。这种方式相对于集中合并具有明显优势，较容易实现负载平衡。同时，通过引入空间分区树，可以保证合并后的局部 Skyline 集合仍然具有相似的投影，有利于提前删除非 Skyline 点。但仍然需要所有的局部 Skyline 点进行两两比较，如果最终的局部 Skyline 集合很大，则代价也非常大，降低这种代价是我们要解决的第二个问题。一般情况下，空间分区机制能够利用以下分析结果。

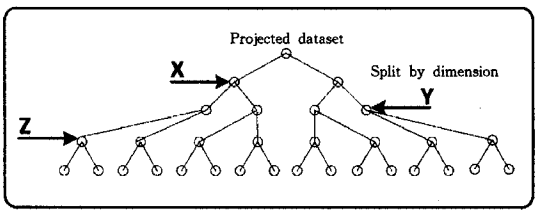

图 3 空间分区树

命题 1 若存在一个数据点 $p=\{p_1,\cdots,p_d\}$，在 x 维上，对于所有 $p_1\in P_1$ 的点都小于 $p_2\in P_2$，于是有：

1)P_1 中没有任何一个点可以控制 P_2 中的点；

2)P_2 中的点控制 P_1 中的点，当且仅当在不包含 x 维坐标的 $d-1$ 维子空间里，P_2 中的点控制(或者与之相同)P_1 中的点。

由于命题 1 的前提条件仅仅是针对超球面投影点，上面的分析结果并不能直接应用到本文提出的分区策略中。因此，P_1 中的点仍然可以控制 P_2 中的点，且在 $d-1$ 维子空间中，P_2 中的点控制 P_1 中的点并不足够成为实际的控制关系。

然而，比较局部 Skyline 点 $p_1 \in P_1$ 和 $p_2 \in P_2$ 时，在 x 维上，p_1 比 p_2 的 x 维值小是不确定的。因此，在 P_2 中，将根据 x 维值排序局部 Skyline 点，然后要比较 p_1 和 p_2 点仅需要确定 $p_1.x \leqslant p_2.x$ 是否成立即可，这很可能大幅度约减比较次数。在实验中，这种启发式方法效果很好。应用这种启发式方法过滤 P_1 中的非 Skyline 点也是可行的。在这里，计算 P_1 中各点的 $d-1$ 维（不包括 x 维）值的和，然后根据和函数值排序 P_1 中的点。这种启发式算法在这里也会起部分作用。

4 实验验证及性能分析

4.1 实验环境和测试数据集

4.1.1 实验环境

这部分将讨论 HSPD-Skyline、MR-Skyline 和 MG-Skyline 算法在 IBM 服务器上获得的实验结果。这部分所用的数据集分别来自 3 个不同分布的人工生成数据集：独立、相关和反相关，如图 4 所示。在实验过程中，由于服务器环境运行着其他服务，本文实验过程是与其他程序共享系统资源，因此，在实验结果上可能会出现小幅波动，但不会影响算法的总体趋势。分布式环境：1 个 Master 管理节点服务器，8 个刀片服务器（HS21）计算节点，一套 8T 磁盘阵列（IBM DS3400）存储，具体软硬件环境配置如下：

1）Master 管理节点服务器：两颗英特尔四核至强 E5420（2.5GHz/EM64T，12MB L2 缓存），配置 8G PC2-5300 CL5 ECC DDR2 667MHz 内存；3 块 146G 硬盘；

2）刀片服务器（HS21）计算节点：两颗四核 Intel Xeon E5430（2.66GHz，12MB L2，1333MHz FSB，80W），8GB（4 * 2GB）PC2-5300 FB-DIMM 内存，1 块 146GB 10k SFF SAS Fixed HDD，双千兆以太网；

3）磁盘阵列：DS3400 磁盘柜 Single Controller，8 * 1T；

4）软件系统：操作系统，RadHat Linux 6.3，64 位；开发环境，Eclipse 及其 Hadoop 开发插件；分布式编程工具，Apache Hadoop；Hadoop-1.0.0。

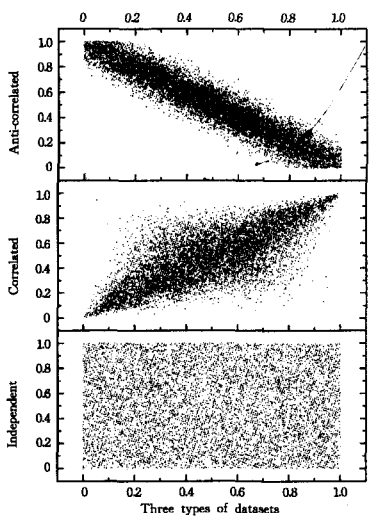

图 4 独立、相关和反相关数据集分布

4.1.2 测试数据集

实验过程中，研究 3 种不同分布的数据集，数据集的生成方式与文献[2]类似，分别为独立、相关和反相关，如果不特别指出，数据范围是[0,1)：

1）独立：独立数据集，使用均匀分布独立生成所有特征属性，图 4 Independent 子图显示 10000 个 2 维的数据点，7 个 Skyline 数据点使用绿色点标识；

2）相关：相关分布数据集（相关度为 0.5），其它维度是维度 1 的单调增函数，图 3 Correlated 子图显示 10000 个 2 维的数据点，3 个 Skyline 数据点使用绿色点标识；

3）反相关：反相关分布数据集（相关度为 −0.5），其他维度是维度 1 的单调减函数，图 3 Anti-correlated 子图显示10000个 2 维的数据点，45 个 Skyline 数据点使用绿色点标识。

使用这 3 类数据集测试 HSPD-Skyline 算法。针对数据集的分布特性、维度、规模等参数，依次运行实验数据集（独立、相关和反相关数据），在此过程中，每次仅改变某个参数，其他参数使用默认设置。

4.2 实验结果与性能分析

4.2.1 Skyline 与数据分布、维度关系

首先对于 Skyline 与数据分布、维度和规模之间的关系进行测试，随机生成的独立、相关和反相关的数据集，通过改变数据集规模和维度，观察数据集的 Skyline 大小，如图 5 所示。图 5(a)示出对于 3 种不同分布的数据集，Skyline 规模随着数据集规模的变化情况。对于 5 维的数据集，随着数据集规模增大，虽然 3 个数据集的 Skyline 规模都在增长，但相关数据集的 Skyline 比例几乎不变，其他两个数据集的 Skyline 比例逐渐降低。图 5(b)示出对于 3 种不同分布的数据集，Skyline 规模随着数据维度的变化情况，对于 10k(10000)的数据集，反相关数据集对于维度的增大变化最为敏感，当维度达到 10 维时，Skyline 集合几乎达到 100%，独立数据集当维度达到 20 维时，Skyline 集合几乎达到 100%，而相关数据集维度到 30 维时，Skyline 集合仅达到 65%。如图 5 所示，数据分布和维度对数据集 Skyline 规模影响较大，数据集规模对其影响较小。由图 5 也可以推测出，随着数据集的相关度的增大，Skyline 的规模也增大，同时 Skyline 查询时间也会变大。

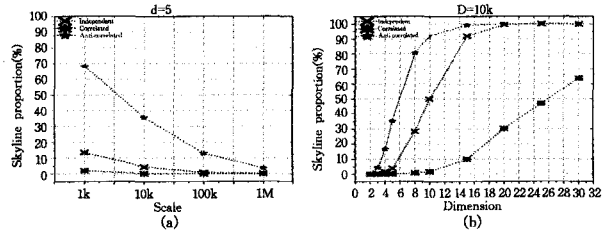

图 5 Skyline 规模随着数据集规模和维度的变化情况

4.2.2 算法性能分析

本文实现了云环境下基于随机分区的 MR-Skyline 和基于格分区的 MG-Skyline 两种算法，设计并实现了基于超球面投影的 Skyline 算法——HSPD-Skyline。对这 3 种算法分别在数据分布、数据规模、维度和计算节点个数进行了大量的实验比较与评估。

对于数据分布分别为独立、相关和反相关情况，评估 HSPD-Skyline、MR-Skyline 和 MG-Skyline 算法的处理时间随着数据规模的变化关系，如图 6 所示。本次实验分别测试了 3 种算法在数据规模 D=1M，10M，100M，1G 个数据点，维度 d=5，计算节点个数 n=8 情况下的运行时间。在数据量较小的情况下，例如 1M 个数据点情况，3 种算法的运行时间相差不大，随着数据规模的增大，运行时间差距慢慢变大，HSPD-Skyline 明显优于 MG-Skyline 和 MR-Skyline，而 MG-

Skyline 优于 MR-Skyline 算法(由于时间度量使用了 Log 评估,这种差距在图 6 上显示不明显,事实上差距较大)。而且,由于引入 Map-Reduce 分布式框架,数据规模呈 10 倍增长,但处理的时间开销增长有限。这与预想是一致的,因为数据规模较小,运行时间主要是 Map-Reduce 框架启动开销和 I/O 开销,Skyline 计算处理开销占总体开销比例不大。因此,HSPD-Skyline 算法的减枝策略和 HA-SPT 启发式策略的优势并没有凸显出来。当数据规模不断变大时,HSPD-Skyline 算法的优势才变得明显,总体运行时间明显优于其他两种方法。

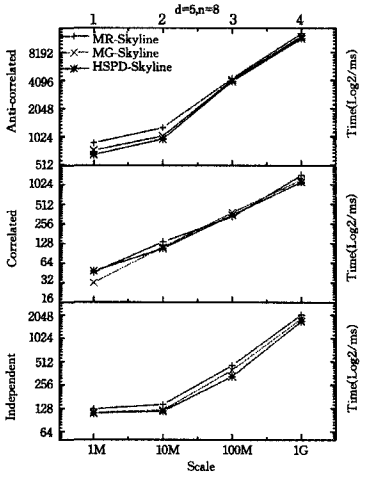

图 6 评估 3 种算法在不同数据分布下运行时间与数据规模的关系

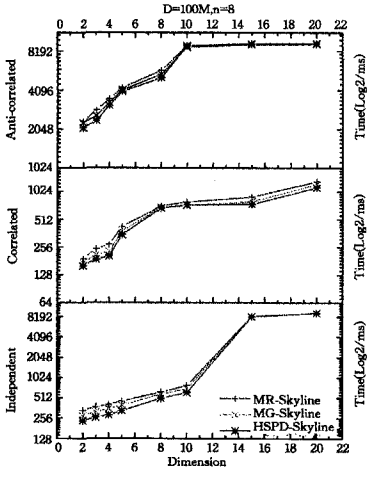

图 7 评估 3 种算法在不同数据分布下运行时间与数据维度的关系

对于数据分布分别为独立、相关和反相关情况,评估 HSPD-Skyline、MR-Skyline 和 MG-Skyline 算法的处理时间随着数据维度的变化关系,如图 7 所示。本次实验分别测试了 3 种算法在数据维度 $d=2,3,4,5,8$ 10,15,20,数据规模 $D=100$M,计算节点个数 $n=8$ 情况下的运行时间。由 4.2.1 节分析得出,3 种不同分布的数据集,对数据维度敏感性不同。反相关数据集对于维度的变化最为敏感,当维度达到 10 维时,Skyline 集合几乎达到 100%,独立数据集当维度达到 20 维时,Skyline 集合几乎达到 100%,而相关数据集维度到 30 维时,Skyline 集合仅达到 65%。图 7 进一步验证了分析的合理性,对于反相关数据集,当维度大于 10 时,数据集的 Skyline 集合接近 100%,3 种算法的运行时间大致相同。而对于独立数据集,当维度为 15 时,3 种算法的运行时间已经开始逐渐接近,为 20 时,3 种算法的运行时间大致相同。由于 3 种分区算法的处理核心为 SFS 算法,因此对于维度变化比较敏感,随着维度升高,算法时间开销不断增大。在一定维度范围内,HSPD-Skyline 算法的时间开销优于 MR-Skyline 和 MG-Skyline 算法。

最后,测试了集群中机器节点数量对算法性能的影响。对于数据分布分别为独立、相关和反相关的情况,评估 HSPD-Skyline、R-Skyline 和 MG-Skyline 算法的处理时间随着数据规模的变化关系,实验结果如图 8 所示。理论上说,充分并行的计算时间花销与计算节点个数成反比,实际情况由于任务分配调度等开销使得多计算节点的优势有所削弱。实验中,使用 5 维、100M 的 3 种分布数据集,测试了节点数为 1~8 个计算节点,并以此推测了在 10、15 和 20 个计算节点上 HSPD-Skyline、MR-Skyline 和 MG-Skyline 算法的运行时间。显然,随着计算节点的增多,算法的总体运行时间呈下降趋势。当计算节点增加到一定程度,对于 3 种算法,I/O 时间和 Map-Reduce 启动时间占了总运行时间的绝大部分,因此继续增加计算节点个数已经几乎不能降低总的时间开销。

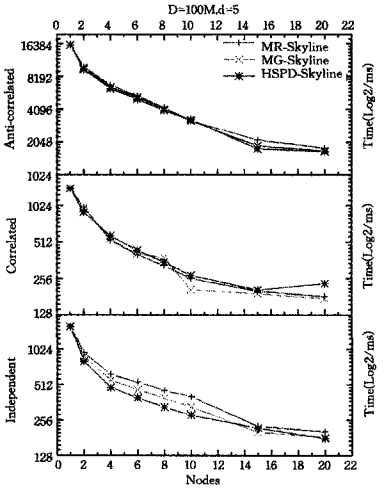

图 8 评估 3 种算法在不同数据分布下运行时间与计算节点数的关系

结束语 本文在 Map-Reduce 框架下实现了基于分区的 Skyline 计算,实现了两种算法:基于随机分区的 Skyline 算法、MR-Skyline、基于格分区的 Skyline 算法、MG-Skyline,设计并实现了基于超球面投影的 Skyline 算法、HSPD-Skyline,并且对这 3 种算法进行了大量的实验比较与评估。HSPD-Skyline 算法在多数情况下运行效率高于 MR-Skyline 和 MG-Skyline 算法,且减枝性能较强,但在分布式环境下依然存在维数灾难现象。

本文工作可以从以下几个方面进行拓展。

1)从分布式计算的角度来考虑,进一步优化 HSPD-Skyline 算法,提高算法的运行效率;

2)虽然 HSPD-Skyline 算法在子分区减枝能力较强,但由于超球面投影过程的计算量比较大,在不考虑空间分区树的启发式策略 HA-SPT 的情况下,HSPD-Skyline 算法的处理效率与 MG-Skyline 算法相近。但由于 MR-Skyline,事实上,是 HA-SPT 策略大大提高了 HSPD-Skyline 算法的时间效率,因此如何设计一个时间复杂度较低且子分区减枝能力较强的分区策略是未来需要研究的方向。

3)在大数据环境下,基于 Map-Reduce 的算法越来越受到重视,因为用户需要在较短的时间内得到分析结果来指导决策。但是 Hadoop 启动 Map-Reduce 的时间开销较大,十几秒到几十秒不等,如何通过优化算法和 Hadoop 环境来提高

时间性能是一个难点。

4)如何将 HSPD-Skyline 算法应用到真实的应用环境来解决实际问题是需要进一步研究的目标。

参 考 文 献

[1] Kung H T, Luccio F, Preparata F P. On finding the maxima of a set of vectors[J]. Journal of the ACM, 1975, 22(4): 469-476

[2] Borzsonyi S, Kossmann D, Stocker K. The skyline operator[C]// Proc. of the 17th Int'l Conf. on Data Engineering. Heidelberg, IEEE Computer Society Press, 2001: 421-430

[3] Chomicki J, Godfrey P, Gryz J, et al. Skyline with presorting[C]// Proc. of the 19th International Conference on Data Engineering (ICDE 2003). 2003: 717-816

[4] Tan K-L, Eng P-K, Ooi B C. Efficient progressive skyline computation[C]// Proc. of the 27th International Conference on Very Large Data Bases(VLDB 2001). 2001: 301-310

[5] Godfrey P, Shipley R, Gryz J. Maximal vector computation in large data sets[C]// Proc. of the 31st international conference on Very large data bases(VLDB 2005). 2005: 229-240

[6] Kossmann D, Ramsak F, Rost S. Shooting stars in the sky: an online algorithm for skyline queries[C]// Proceedings of the 28th International Conference on Very Large Data Bases. 2002: 275-286

[7] 周红福,宫学庆,郑凯,等. 基于高维空间的在线高效子空间 Skyline 算法-CSky[J]. 计算机学报,2007,30(8):1409-1417

[8] 程文聪,邹鹏,贾焰,等. 基于小波概要的区间差分 skyline 研究[J]. 计算机科学,2010,37(11):160-165,202

[9] 付世昌,董一鸿,陈华辉,等. 基于道路网络不确定移动对象的连续概率 Skyline 查询[J]. 计算机科学,2011,38(7):152-156

[10] Balke W-T, Güntzer U, Zheng J X. Efficient distributed skylining for web information systems[C]// Proc. of the 9th International Conference on Extending Database Technology (EDBT 2004). 2004: 256-273

[11] Huang Zhi-yong, Jensen C S, Lu Hua, et al. Skyline queries against mobile lightweight devices in manets[C]// Proc. of the 22nd International Conference on Data Engineering (ICDE 2006). 2006: 66-77

[12] Lo E, Yip K Y, Lin K-I, et al. Progressive skylining over web-accessible databases[J]. Data & Knowledge Engineering, 2006, 57(2): 122-147

[13] Vlachou A, Doulkeridis C, Kotidis Y, et al. Skypeer: Efficient subspace skyline computation over distributed data[C]// Proc. of the IEEE 23rd International Conference on Data Enginering (ICDE 2007). 2007: 416-425

[14] Jagadish H V, Ooi B C, Vu Q H. Baton: a balanced tree structure for peer-to-peer networks[C]// Proc. of the 31st international conference on Very large data bases(VLDB 2005). 2005: 661-672

[15] Wu Ping, Zhang Cai-jie, Feng Ying, et al. Parallelizing skyline queries for scalable distribution[C]// Proc. of the 10th International Conference on Extending Database Technology (EDBT 2006). 2006: 112-130

[16] Gao Yun-jun, Chen Gen-cai, Chen Ling, et al. Parallelizing progressive computation for skyline queries in multi-disk environment[C]// Proc. of the 17th International Conference on the Database and Expert Systems Applications (DEXA 2006). 2006: 697-706

[17] Ratnasamy S, Francis P, Handley M, et al. A scalable content addressable network[C]// Proc. of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications(SIGCOMM 2001). 2001: 161-172

[18] Pei Jian, Yuan Yi-dong, Lin Xue-min, et al. Towards multidimensional subspace skyline analysis[J]. ACM Trans. on Database Systems (TODS), 2006, 31(4): 1335-1381

[19] DeWitt D, Gray J. Parallel database systems: the future of high performance database systems[J]. Commun. ACM, 1992, 35(6): 85-98

[20] Stonebraker M, Aoki P M, Litwin W, et al. Mariposa: a wide-area distributed database system[J]. The VLDB Journal, 1996, 5(1): 048-063

[21] Dehne F, Fabri A, Rau-Chaplin A. Scalable parallel geometric algorithms for coarse grained multicomputers[C]// Proc. of the Ninth Annual Symposium on Computational Geometry (SCG 1993). 1993: 298-307

[22] Cosgaya-Lozano A, Rau-Chaplin A, Zeh N. Parallel computation of skyline queries[C]// Proc. of the 21st International Symposium on High Performance Computing Systems and Applications (HPCS 2007). 2007, 12

[23] Kung H T T, Luccio F L, Preparata F P. On finding the maxima of a set of vectors[J]. Journal of the ACM (JACM), 1975, 22(4): 469-476

[24] Papadias D, Tao Yu-fei, Fu G, et al. An optimal and progressive algorithm for skyline queries[C]// Proc. of the 2003 ACM SIGMOD International Conference on Management of Data (SIGMOD 2003). 2003: 467-478

[25] Huang Zhi-yong, Jensen C S, Lu Hua, et al. Skyline queries against mobile lightweight devices in manets[C]// Proc. of the 22nd International Conference on Data Engineering (ICDE 2006). 2006, 66

[26] Wang Shi-yuan, Ooi B C, Tung A K H. Efficient skyline query processing on peer-to-peer networks[C]// Proc. of the IEEE 23rd International Conference on Data Enginering(ICDE 2007). 2007: 1126-1135

[27] Cosgaya-Lozano A, Rau-Chaplin A, Zeh N. Parallel computation of skyline queries[C]// Proc. of Int. Symp. on High Performance Computing Systems and Applications. 2007, 12

[28] Wu P, Zhang C, Feng Y, et al. Parallelizing skyline queries for scalable distribution[C]// Proc. of Conf. on Extending Database Technology (EDBT). 2006: 112-130

[29] Wang S, Ooi B C, Tung A K H, et al. Efficient skyline query processing on peer-to-peer networks[C]// Proc. of Int. Conf. on Data Engineering (ICDE). 2007: 1126-1135

[30] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[J]. Commun. ACM, 2008, 51: 107-113

[31] 丁琳琳,信俊昌,王国仁,等. 基于 Map-Reduce 的大数据高效 Skyline 查询处理[J]. 计算机学报,2011,34(10):1785-1796

[32] 张波良,周水庚,关佶红. MapReduce 框架下的 Skyline 计算[J]. 计算机科学与探索,2011,5(5):385-397

[33] Vlachou A, Doulkeridis C, Kotidis Y. Angle-based space partitioning for efficient parallel skyline computation[C]// SIGMOD. 2008: 227-238

[34] Köhler H, Yang Jing, Zhou Xiao-fang. Efficient parallel skyline processing using hyperplane projections[C]// Proceedings of the 2011 International Conference on Management of Data. 2011: 12-16