基于 Map-Reduce 的海量数据高效 Skyline 查询处理

丁琳琳 信俊昌 王国仁 黄 山
(医学影像计算教育部重点实验室(东北大学) 沈阳 110819)
(东北大学信息科学与工程学院 沈阳 110819)

摘 要 Skyline 查询已成为现今数据库和信息检索领域的研究热点之一,伴随着人类可以采集和利用的数据信息的急剧增长,使得如何处理海量数据的 Skyline 查询成为急需解决的问题. 近年来兴起的 Map-Reduce 编程框架能够有效地处理基于海量数据的应用,该文既是研究如何运用 Map-Reduce 编程框架解决海量数据的 Skyline 查询问题. 在 Map-Reduce 框架下处理 Skyline 查询的直接方法是扫描整个数据集进而得到查询结果,但是在海量数据 Skyline 查询问题中,查询结果的数量远小于原始数据集的数据量,对此该文提出了一系列的 Skyline 查询算法及优化,有效地过滤掉部分不能成为 Skyline 查询结果的数据对象,大幅度提高了在 Map-Reduce 框架下处理 Skyline 查询的效率. 大量运行在 Hadoop 平台上的实验验证了该文所提出的 Skyline 查询处理算法具有良好的有效性、准确性和可用性.

关键词 云计算;Skyline 查询;Map-Reduce;海量数据;Hadoop
中图法分类号 TP311 **DOI 号**: 10.3724/SP.J.1016.2011.01785

Efficient Skyline Query Processing of Massive Data Based on Map-Reduce

DING Lin-Lin XIN Jun-Chang WANG Guo-Ren HUANG Shan
(*Key Laboratory of Medical Image Computing of Ministry of Education (Northeastern University), Shenyang* 110819)
(*College of Information Science and Engineering, Northeastern University, Shenyang* 110819)

Abstract Recently, Skyline query has been a research hot of Database and Information Retrieval. In addition, the amount of data for collecting and using by human is developing at an astonishing speed. Therefore, how to process Skyline query of massive data is an urgent problem. Map-Reduce is a new parallel programming model that processes vast number of data on large clusters with easy deployment. As a parallel programming model, Map-Reduce is suit for solving Skyline query of massive data. This paper resolves the problem of processing Skyline query of massive data on Map-Reduce framework. A straightforward implementation of Skyline query on Map-Reduce needs to scan all the candidate results before obtaining the final results. However, when the amount of final results is much smaller than the original data, there is a waste of processing unnecessary results on Map-Reduce framework. Consequently, in this paper, a series of efficient Skyline query algorithms and optimization have been proposed to prune the unpromising results effectively and enhance the performance of processing Skyline query of massive data on Map-Reduce. Our extensive experiments are built on top of Hadoop platform, an open-source implementation of Map-Reduce framework. The experiment results demonstrate that our algorithms have high efficiency, accuracy and scalability.

Keywords cloud computing; skyline query; Map-Reduce; massive data; hadoop

收稿日期:2011-08-13. 最终修改稿收到日期:2011-09-15. 本课题得到国家自然科学基金重点项目(60933001)、国家杰出青年科学基金(61025007)及中央高校基本科研业务费专项基金(N090304007)资助.**丁琳琳**,女,1983 年生,博士,主要研究方向为云数据管理和 P2P 数据管理. E-mail: linlin.neu@gmail.com. **信俊昌**,男,1977 年生,博士,讲师,主要研究方向为感知数据和不确定数据. **王国仁**(通信作者),男,1966 年生,教授,博士生导师,主要研究方向为 XML 数据管理、生物信息学、分布与并行数据库、多媒体索引技术、并行计算等. E-mail: wanggr@mail.neu.edu.cn. **黄 山**,男,1986 年生,博士研究生,主要研究方向为云数据管理.

1 引 言

随着信息时代的到来,Skyline 查询已成为现今数据库和信息检索领域的研究热点之一,并被广泛地应用于多目标决策分析和数据可视化等领域. Skyline 查询的主要目标是在给定的数据集中搜索不被其它数据对象“支配”的数据对象. 一个数据对象“支配”另一个数据对象指的是该数据对象在所有维度都不比另一个数据对象“差”,并且至少有一个维度比另一个数据对象“优”,这里“优”和“差”的定义并无统一标准,而是根据不同应用的特点确定. 例如,图 1 所示为某沿海城市酒店的 Skyline 查询情况. 酒店的价格和到海边的距离是衡量某个酒店“优”或“差”的两个标准,可以看出到海边距离越近的酒店其价格也越贵,反之,酒店价格越便宜,在图 1中位于虚线上的点既是该 Skyline 查询的结果集,在结果集中的点即为不被其它点所“支配”的点. 当游客到该城市旅行,希望找到既离海边近并且价格又便宜的酒店时,只需在该 Skyline 查询结果集中的点进行选择,就可以找到令游客满意的酒店. 除此之外,Skyline 查询还可以应用到如何选择工作能力强且要求薪水低的员工和如何买到质量好并且价格低的商品中.

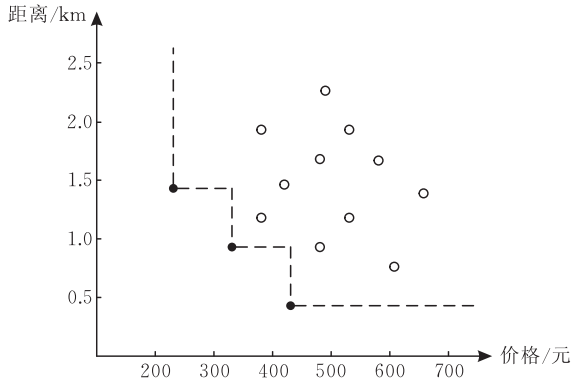


图 1 酒店 Skyline 示意图

然而,随着网络信息和通信技术的迅猛发展,人类可以采集并利用的数据量急剧增长,使得如何在海量数据集中迅速确定 Skyline 查询结果成为迫切需要解决的问题. 自 2001 年,Borzsonyi 等人[1]将 Skyline 查询引入到数据库中以来,其得到了广泛的重视并取得了一系列的研究成果[2]. 尽管,现有的 Skyline 查询能够解决很多实际问题,但这些算法并不适用于海量数据的处理. 近年来出现的 Map-Reduce[3]并行编程框架定位于处理海量数据,具有良好的可扩展性和容错性,能够满足数据量迅速增长的需要,因此可以运用 Map-Reduce 框架解决海量数据的 Skyline 查询处理问题. 本文主要研究如何在 Map-Reduce 框架下处理 Skyline 查询. 在分析了基于 Map-Reduce 框架处理 Skyline 查询的基本算法后,针对 Skyline 查询结果的数量远小于原始数据集的数据量的问题,本文又提出了一系列的 Skyline 查询算法,能够有效地过滤部分不能成为查询结果的数据对象,减少 Map 任务的输出,同时降低 Reduce 任务的输入量,使得在有效解决海量数据 Skyline 查询问题的同时提高了算法的效率,降低了网络开销.

本文主要贡献点如下:

(1) 提出 3 个在 Map-Reduce 框架下具有过滤性质的海量数据高效 Skyline 查询处理算法,分别是延迟 Skyline 查询算法、贪婪 Skyline 查询算法和混合 Skyline 查询算法.

(2) 提出 1 个 Skyline 查询优化算法,前置 Skyline 查询算法.

(3) 大量实验结果验证了本文提出的 Skyline 算法具有高效、准确和可扩展等特性.

本文第 2 节介绍相关工作;第 3 节提出 Map-Reduce 框架下海量数据 Skyline 查询处理算法;第 4 节提出 Skyline 查询处理优化算法;第 5 节分析实验结果;第 6 节进行总结.

2 相关工作

2.1 Skyline 查询

目前,Skyline 查询的相关研究主要分为集中式 Skyline 查询和分布式 Skyline 查询. 集中式 Skyline 查询主要针对数据量大且更新不频繁的情况. 集中式 Skyline 查询算法中的 BNL 算法[1]的基本原理是通过将数据集中某个数据对象与数据集中其它数据对象比较用以检索数据集中不被其它数据对象支配的数据对象. Bitmap 算法[4]是一种利用比特位运算的高效方法,利用位运算快的特点提高算法的效率. NN 算法[5]的基本原理是先对数据集中的数据对象构建 R-tree 索引,并利用最近邻算法求出距离原点最近的点,然后利用该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,直到分区中不含有任何 Skyline 查询结果为止. 文献[6]提出了一种在线高效子空间 Skyline 算法 CSky,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. 分布式 Skyline 查询算法的研究主要集中于算法的渐进性，节点之间负载平衡等，用于万维网、P2P 等网络中. BDS 算法[7]的基本原理是先确定数据集中一个包含 Skyline 查询结果和非查询结果的数据对象的子集，并从中删除非查询结果的数据对象，形成最终 Skyline 查询结果. IDS 算法[7]的基本原理是利用启发式规则减少候选集合计算中的访问次数，从而在分布式环境中提高系统的效率. 文献[8]提出了一种基于 P2P 网络的 CAN 结构的分布式 Skyline 查询算法 DSL，主要原理是采用负载平衡策略在节点间分割数据并采用动态编码策略提高了 Skyline 查询的效率. 文献[9]提出一种有效的滑动窗口 Skyline 监督算法及其优化，通过采用不同的过滤方法减少传输的数据量，降低了算法的代价. 不论是集中式 Skyline 查询算法还是分布式 Skyline 查询算法，面对日益增长的海量数据，都会增加处理 Skyline 查询的计算量和代价，降低了 Skyline 查询算法的性能.

2.2　Map-Reduce 框架

Map-Reduce 是现今较为流行的并行编程框架，用于大规模数据集的并行计算. 最早是由 Dean 等人在 2004 年提出的，其开源实现是 Hadoop 平台①. 图 2 说明了 Map-Reduce 的基本框架，框架中包含一个 Master 节点和多个 Slave 节点，Master 节点负责整个框架的任务调度和分配，Slave 节点负责执行分配到自身的任务. 在 Map-Reduce 框架中一个作业的处理过程主要包括两个阶段：用于划分原始数据和产生中间结果的 Map 阶段和生成最终结果的 Reduce 阶段. 用户可以通过简单的配置 Mapper 函数和 Reducer 函数来实现对原有任务的并行执行. 近年来，关于 Map-Reduce 的研究也不断深入，涌现出一系列的研究成果，主要围绕在改进 Map-Reduce 架构以适应处理某些特定应用和将某些现有问题在 Map-Reduce 框架下处理两方面. Hadoop++[10]通过调用用户自定义函数提高 Map-Reduce 的性能，但并未改变 Hadoop 原有的框架. HaLoop[11]是 Hadoop 的改进版本，通过增加和修改一些功能模块有效地处理具有迭代特征的应用. Map-Reduce-Merge[12]通过给 Map-Reduce 框架增加一个 Merge 阶段有效地合并已经被 Map 和 Reduce 阶段划分和排序的数据，用来处理关系数据库的相关应用. 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 查询的相关研究还相当初步. 目前，文献[13-14]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（Map-Reduce based Block-Nested-Loops，MR-BNL）等 3 个 Skyline 查询算法. 文献[13-14]仅仅是在 Map-Reduce 框架下实现了 Skyline 查询处理算法，并未对算法提出优化，而本文针对海量数据的 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提出了一系列优化算法，提高了在 Map-Reduce 框架下处理 Skyline 查询问题的效率.

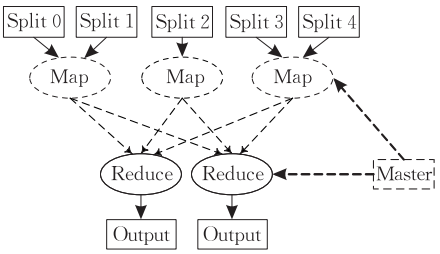


图 2　Map-Reduce 框架

3　基于 Map-Reduce 框架的高效 Skyline 查询处理算法

本节主要说明 Map-Reduce 框架下 Skyline 查询处理算法. 首先，3.1 小节介绍了 Map-Reduce 框架下 Skyline 查询的基本算法，3.2～3.4 小节分别介绍了本文所提出的 3 个 Skyline 查询算法，分别是延迟 Skyline 查询算法、贪婪 Skyline 查询算法和混合 Skyline 查询算法.

3.1　基本 Skyline 查询算法

文献[13-14]中所提出的基于 Map-Reduce 的块嵌套循环算法（Map-Reduce based Block-Nested-Loops，MR-BNL）、基于 Map-Reduce 的排序过滤算法（Map-Reduce based Sort-Filter-Skyline，MR-SFS），其执行过程及原理可以归纳为在 Map-Reduce 框架下处理 Skyline 查询的基本算法，本文选择性能略优的 MR-BNL 算法为例进行分析和说明. 下面以一个二维空间 Skyline 查询为例说明基本 Skyline 查询算法的原理. 如图 3 所示，在二维空间

① Hadoop. http://hadoop.apache.org/

中有 50 个数据点，求这 50 个点的 Skyline 查询结果，由图 3 可知实心圆点表示该 Skyline 查询的结果即点(1,9),(2,5),(3,3),(4,2),(5,1). 假设整个 Map-Reduce 框架中包含 5 个 Map 任务和 1 个 Reduce 任务，50 个数据点被随机划分成 5 个分片，分别由 5 个 Map 任务执行，得到自身的中间 Skyline 结果，进行排序和分组后发送给 Reduce 任务. Reduce 任务接收到所有 Map 任务的中间结果后产生最终的 Skyline 查询结果，具体执行过程如图 4 所示. 例如，对于第 1 个 Map 任务来说，原始数据点为(2,10),(4,10),(1,9),(2,9),(4,9),(3,8),(5,8),(2,7),(4,7),(2,5)，产生的 Skyline 查询结果为点(1,9)和(2,5)，同理，第 2 个 Map 任务产生的 Skyline 查询结果为点(5,9),(6,8)和(8,7)，第 3、4、5 个 Map 任务产生的 Skyline 查询中间结果集为点(3,3),(4,2),(5,5),(7,2),(5,1). Reduce 任务接收到 5 个 Map 任务所产生的中间结果之后，通过调用用户定义的 Reducer 函数求得该 Skyline 查询的最终结果，即点(1,9),(2,5),(3,3),(4,2),(5,1).

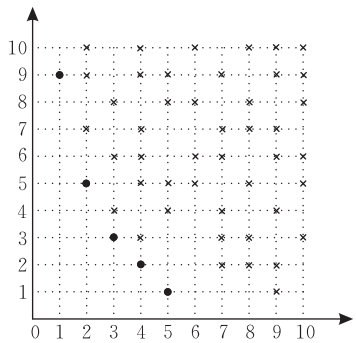


图 3 Skyline 数据空间

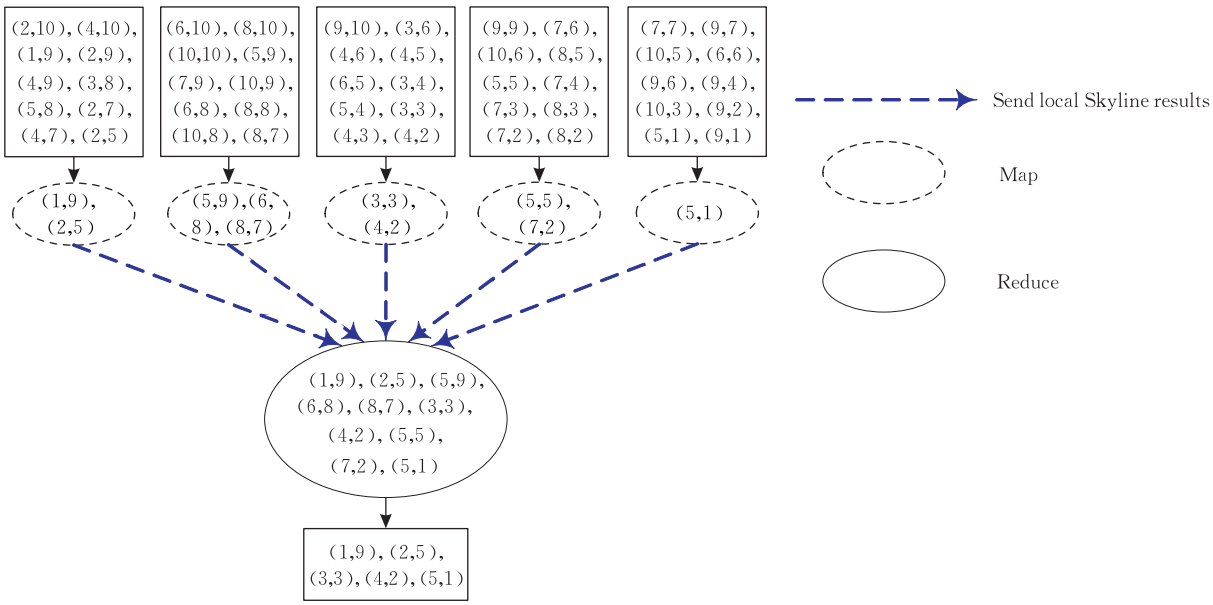


图 4 基本 Skyline 算法

以上简要介绍了 Map-Reduce 框架下 Skyline 查询处理的基本算法，可以看出通常情况下 Skyline 查询最终结果的数量远小于原始数据集的数据量，即使采用 Map-Reduce 框架将原有数据集划分成多个子数据集进行并行计算，其代价仍然很大，在数据量成指数增长的情况下更会使效率降低. 为了进一步提高 Skyline 查询的效率，本文在用 Map-Reduce 框架处理 Skyline 查询时，通过采用一系列有效的方法过滤掉一部分 Map 任务的输出结果，即不是最终查询结果的数据对象，使得 Reduce 任务的输入量大幅度减少，提高了运用 Map-Reduce 框架处理 Skyline 查询的效率. 具体过程如下：在每个 Map 任务结束后，通过与 Master 节点建立连接，每个 Map 任务选择一个自身的 Skyline 查询结果作为其局部过滤值发送给 Master 节点，Master 节点根据不同的策略在接收到的局部过滤值中选择一个作为全局过滤值，然后 Map 任务根据全局过滤值对自身 Skyline 查询结果进行过滤，过滤一些不能成为 Skyline 查询结果的数据对象，再发送给 Reduce 任务，使得输入量大大减少. Reduce 任务接收过滤后的中间结果，然后调用用户定义的 Reducer 函数产生最终的 Skyline 查询结果.

3.2 延迟 Skyline 查询算法

延迟 Skyline 查询算法的基本原理是：在所有 Map 任务完成后将自身的局部过滤值发送给 Master 节点，Master 节点在接收到所有 Map 任务的局

部过滤值之后产生全局过滤值再将其发送给每个 Map 任务，每个 Map 任务接收到全局过滤值之后，运用该全局过滤值对自身结果进行过滤并产生各自 Skyline 查询结果，Reduce 任务以过滤后的中间结果为输入产生最终的 Skyline 查询结果.

图 5 表示的是运用延迟 Skyline 查询算法处理 3.1 节中 Skyline 查询的过程. 原有数据集被划分成 5 个分片，分别由 5 个 Map 任务处理，每个 Map 任务处理自身的数据得到局部 Skyline 查询结果，并产生局部过滤值. 本文采用文献[9]中的滑动窗口 Skyline 监督算法(SWSMA)求得局部过滤值和全局过滤值，基本原理是：假设集合 S 是 D 维空间中的一个数据集，其中第 i 维的范围是$[0,D_i]$，X 和 Y 是数据集 S 中的两个数据对象，分别表示成$(x_1,x_2,\cdots,x_D)$和$(y_1,y_2,\cdots,y_D)$，数据对象 X 的概率密度函数为 $p(x)=p(x_1,x_2,\cdots,x_D)$，则数据对象 Y 可以“支配”的数据对象数量由下面的式(1)确定：

$$c(Y)=|S|\times\int_R p(x_1,x_2,\cdots,x_D)\mathrm{d}x_1\mathrm{d}x_2\cdots\mathrm{d}x_D \tag{1}$$

其中 $R=\{X|x_1\geq y_1,x_2\geq y_2,\cdots,x_D\geq y_D\}$. 因此，对于每个数据对象来说，由式(1)求得的值越大，表明其过滤能力越强，“支配”能力也越强. 尽管式(1)中的 $p(x)$在大多数情况下并不已知，但可以运用随机取样或直方图的方法近似求得. 另外，求得局部过滤值和全局过滤值的方法不仅限于本文中的一种，任何求 Skyline 查询点的方法均可以用来求局部过滤值和全局过滤值. 以二维空间为例，对于每个数据点来说，可以运用文献[9]中的方法确定该点的“支配”能力，凡是位于该点“支配”范围内的点，均是被该点“支配”的点，即一定不是 Skyline 查询结果. 运用这种方法，每个 Map 任务对自身局部 Skyline 查询结果比较其“支配”能力，并将具有最大“支配”能力的点作为其局部过滤值，发送给 Master 节点. 在图 5 中，本文仅以 3 个步骤进行说明.

步 1. 每个 Map 任务在产生局部过滤值后将其发送给 Master 节点. 例如，第 1 个 Map 任务产生的 Skyline 查询结果为点(1,9)，(2,5)，而点(2,5)的“支配”能力大于点(1,9)，即点(2,5)可以“支配”的范围大于点(1,9)，因此，第 1 个 Map 任务产生的局部过滤值为点(2,5). 同理，第 2～5 个 Map 任务产生的局部过滤值分别为点(6,8)，(3,3)，(5,5)，(5,1)；

步 2. Master 节点接收到所有 Map 任务传送的局部过滤值，在这些局部过滤值中运用同样的方法确定“支配”能力最强的点，这里为点(3,3)，作为全局过滤值，然后将点(3,3)发送给所有 Map 任务；

步 3. 在每个 Map 任务接收到全局过滤值即点(3,3)之后，用其对自身的 Skyline 结果进行过滤，产生最终的 Map 任务输出. 例如第 2 个 Map 任务在接收到点(3,3)之后，用点(3,3)对其 Skyline 查询结果点(5,9)，(8,7)，(6,8)进行过滤，点(3,3)可以“支配”这 3 个点，故这 3 个点一定不是最终的 Skyline 查询结果，即第 2 个 Map 任务的最终输出结果为空. 同理，第 4 个 Map 任务也用点(3,3)对其结果点(5,5)，(7,2)进行过滤，点(3,3)可以支配点(5,5)，但不能支配点(7,2)，故第 4 个 Map 任务最终的输出结果为点(7,2). 因此，Reduce 任务的输入为点(1,9)，(2,5)，(3,3)，(4,2)，(7,2)，(5,1)，与基本 Skyline 算法相比降低了 40%，大幅度减少了 Reduce 任务的输入，因此能够提高整个 Skyline 查询算法的效率.

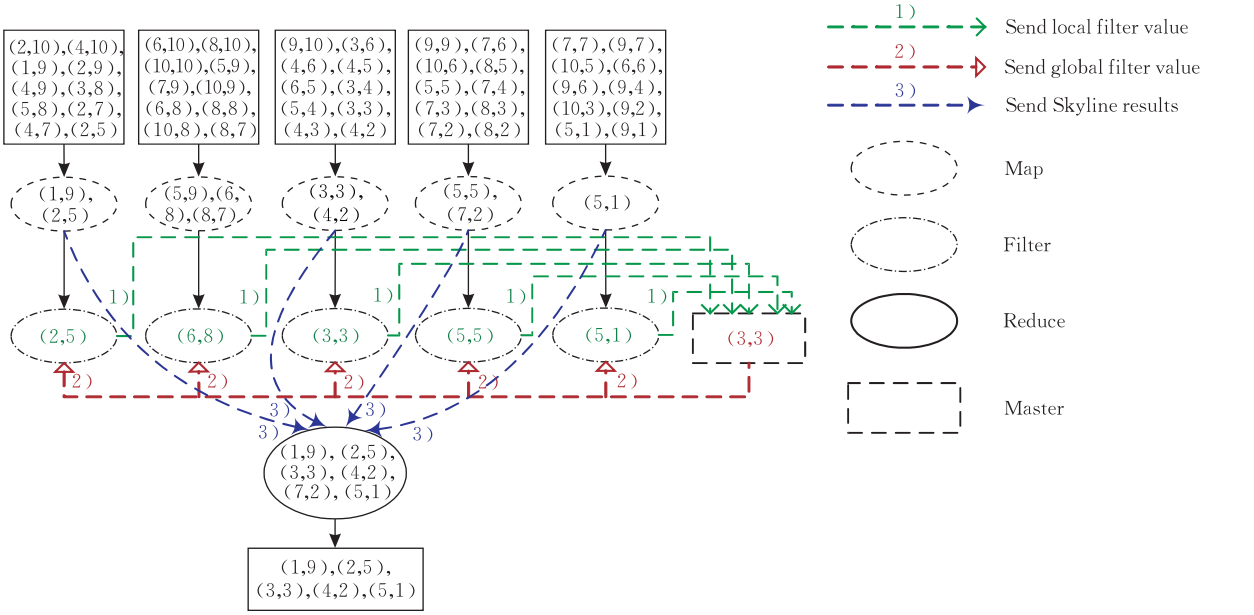


图 5　延迟 Skyline 算法

然而，对于延迟 Skyline 查询算法来说，如果系统中某些 Map 任务占据的时间较长，那么 Master 节点必须等待这些 Map 任务结束才能产生全局过滤值. 在实际应用中，系统中运行的 Map 任务数量巨大，如果某个 Map 任务失效或占据时间较长，会导致该 Map 任务不能及时将局部过滤值发送给 Master 节点的情况，使得 Master 节点由于一直等待该 Map 任务发送局部过滤值而无法产生全局过滤值，系统会进入“死锁”状态，那么 Master 节点无法等待所有 Map 任务结束后再产生全局过滤值. 因此，尽管延迟 Skyline 算法可以降低 Reduce 任务的输入量，但是其效率并不高，并且实际应用的情况很少.

3.3 贪婪 Skyline 查询算法

贪婪 Skyline 查询算法的基本原理是：当一个 Map 任务结束后将其局部过滤值发送给 Master 节点并获得一个全局过滤值. 如果这个全局过滤值为空，那么该 Map 任务将其所有 Skyline 查询结果输出，否则，该 Map 任务用这个从 Master 节点获得的全局过滤值对自身 Skyline 查询结果进行过滤. Master 节点在接收到第一个 Map 任务的局部过滤值后将其设定为全局过滤值. 如果系统中另外一个 Map 任务结束，该 Map 任务也将其局部过滤值发送给 Master 节点. 当 Master 节点接收到该 Map 任务的局部过滤值之后，将其与原有全局过滤值比较，根据不同应用的需求选择最优的值作为新的全局过滤值. 这样的过程一直继续，直到所有的 Map 任务运行结束为止.

图 6 表示贪婪 Skyline 查询算法的计算过程. 与前述 Skyline 查询处理算法相同，原始数据集被随机划分成 5 个分片，分别由 5 个 Map 任务执行，图 6 仅用 6 个步骤说明第 1 和第 2 个 Map 任务的执行情况，其余 Map 任务的执行原理与前 2 个 Map 任务相同，故不再一一列举.

步 1. 当系统中第 1 个 Map 任务结束后，产生 Skyline 查询结果为点(1,9)和(2,5)，其中点(2,5)作为其局部过滤值被发送给 Master 节点；

步 2. Master 节点接收到点(2,5)后即将其作为当前的全局过滤值，发送给第 1 个 Map 任务；

步 3. 第 1 个 Map 任务用点(2,5)过滤其 Skyline 查询结果，过滤后结果仍为点(1,9)和(2,5)，则将点(1,9)和(2,5)作为其最终的 Skyline 查询结果发送给 Reduce 任务；

步 4. 如果第 2 个 Map 任务运行结束，将其局部过滤值点(6,8)发送给 Master 节点；

步 5. Master 节点在接收到点(6,8)之后，将点(6,8)与当前全局过滤值点(2,5)做比较，得到最优值点(2,5)，将点(2,5)作为当前最新的全局过滤值；

步 6. 第 2 个 Map 任务在接收到全局过滤值点(2,5)之后，用其对自身 Skyline 查询结果进行过滤，点(2,5)能“支配”其所有结果点，即第 2 个 Map 任务最终的输出结果为空，故图 6 中并未画出. 这样的过程一直继续直到第 5 个 Map 任务结束，求得最终 Skyline 查询结果.

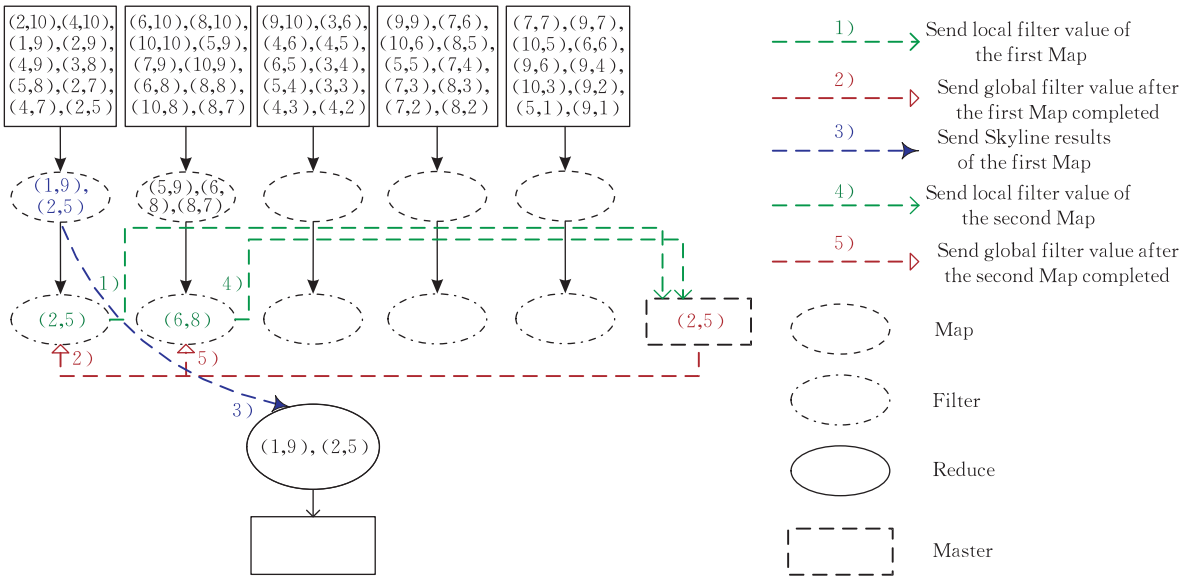


图 6 贪婪 Skyline 算法

与延迟 Skyline 查询算法相比，贪婪 Skyline 查询算法同样能将 Reduce 任务的输入降低到基本 Skyline 算法的 60%，并且不需要等待系统中所有 Map 任务都结束才能产生全局过滤值，但是每当有

一个 Map 任务完成，就要向 Master 节点发送局部过滤值，导致当前全局过滤值不一定是最优全局过滤值，所以本文又提出了混合 Skyline 查询算法来优化延迟 Skyline 和贪婪 Skyline 查询算法.

3.4　混合 Skyline 查询算法

混合 Skyline 查询算法的基本原理是：当系统中的一个 Map 任务运行结束后立即将其局部过滤值发送给 Master 节点，然而，Master 节点并不立刻产生全局过滤值发送给该 Map 任务，而是在一个预先定义的时间间隔内等待其它 Map 任务的完成，Master 节点收集到这段时间间隔内所有完成的 Map 任务的局部过滤值，从中选择最优值作为全局过滤值，并发送给这些 Map 任务. Map 任务在接收到全局过滤值之后，用其过滤自身的 Skyline 查询结果，并生成最终的局部 Skyline 查询结果. 同样，在下一个时间间隔内，Master 节点仍然等待在这个时间间隔内完成的所有 Map 任务的局部过滤值，从中选择最优值作为全局过滤值，并与原有的全局过滤值比较，再从中选择最优值作为新的全局过滤值. 这样的过程一直继续直到所有的 Map 任务结束为止. 当系统中最后一个 Map 任务结束后，它将其局部过滤值发送给 Master 节点，会立即获得一个全局过滤值.

图 7 仍然以相同的 Skyline 查询为例，说明混合 Skyline 查询算法的过程. 原有数据集同样被划分成 5 个分片，分别由 5 个 Map 任务执行，为了描述方便，在混合 Skyline 算法中，本文设定每个时间间隔内共有 2 个 Map 任务完成，本文用 6 个步骤说明前 2 个时间间隔内 Map 任务的执行情况.

步 1. 当第 1 个 Map 任务完成后，将其局部过滤值点(2,5)发送给 Master 节点，Master 节点在接收到点(2,5)后，并不立即产生全局过滤值，而是等待其它 Map 任务的完成. 在这个时间间隔内，第 2 个 Map 任务完成后将其局部过滤值点(6,8)发送给 Master 节点；

步 2. Master 节点比较点(2,5)和点(6,8)，从中选择最优值点(2,5)作为当前的全局过滤值，并发送给第 1 和第 2 个 Map 任务；

步 3. 第 1 个 Map 任务的 Skyline 查询结果不能被点(2,5)“支配”，故其最终的 Skyline 查询结果为点(1,9)，(2,5). 而第 2 个 Map 任务的 3 个 Skyline 查询结果都能被点(2,5)“支配”，所以第 2 个 Map 任务的输出结果为空；

步 4. 在下一个时间间隔内，第 3 个 Map 任务完成后将其局部过滤值点(3,3)发送给 Master 节点，同样，Master 节点也不立即产生新的全局过滤值，而是等待其它 Map 任务的完成，当第 4 个 Map 任务完成后，将其局部过滤值点(5,5)发送给 Master 节点；

步 5. Master 节点从点(3,3)和点(5,5)中选择点(3,3)作为临时全局过滤值，并将其与原有的全局过滤值点(2,5)比较，从中选择最优值点(3,3)作为新的全局过滤值，并发送给这些 Map 任务；

步 6. 在接收到全局过滤值点(3,3)后，第 3、4 个 Map 任务用其对自身结果进行过滤，发现点(3,3)可以“支配”点(5,5)，故第 4 个 Map 任务最终的 Skyline 结果为点(7,2). 整个过程一直继续直到所有 Map 任务都结束.

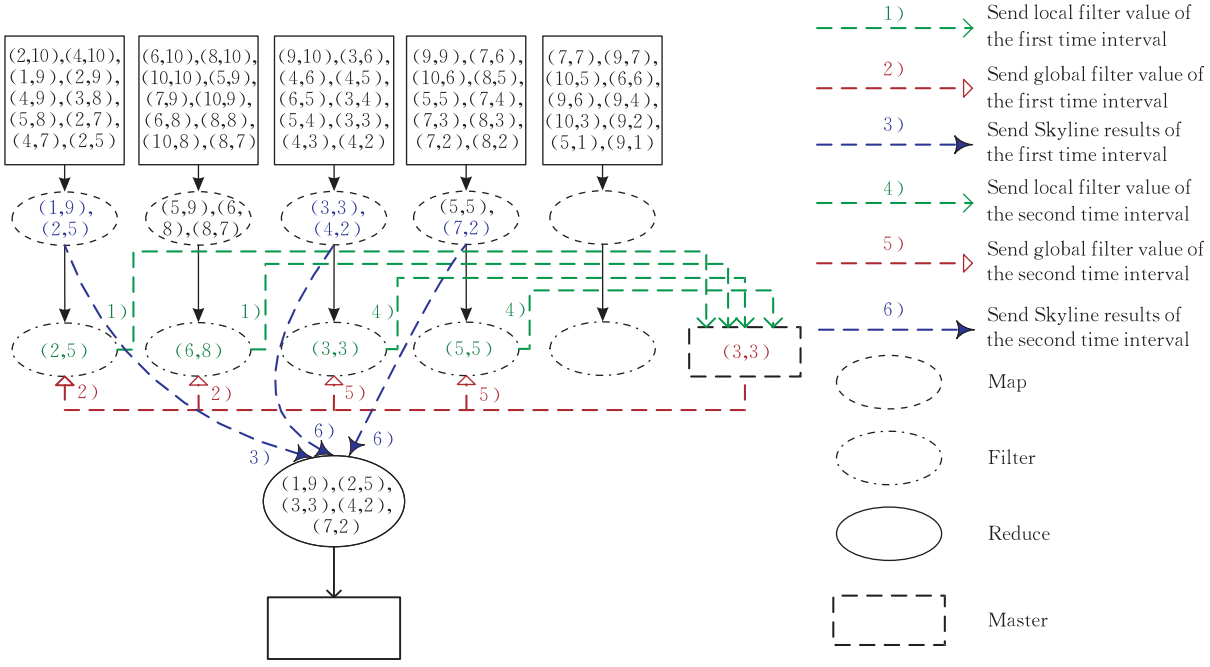


图 7　混合 Skyline 算法

混合 Skyline 查询算法有效地继承了延迟 Skyline 查询算法和贪婪 Skyline 查询算法的优点，与基本 Skyline 算法相比，Map 任务的输出同样能够降低 40%. 混合 Skyline 算法既不用等待系统所有 Map 任务完成才能产生全局过滤值，也不用每次在 Map 任务完成后都与 Master 节点进行通信，在减少 Reduce 任务输入的同时提高了算法的效率，降低了算法的开销.

4 Skyline 查询处理优化算法

在实际应用中，Map 任务的数量远比虚拟节点的数量多很多，甚至能达到几百倍，尽管本文提出了上述 3 个 Skyline 查询算法，但对于海量数据的 Skyline 查询处理的性能仍需进一步提高. 如果能让 Map 任务在运行之前可以预先过滤掉一些不是最终结果的数据，那么 Map 任务的输入数据量也会大大减少，这样会进一步提高 Skyline 查询算法的效率. 因此，本文提出了一个 Skyline 查询优化算法，称为前置 Skyline 查询算法，能够更有效地处理 Map-Reduce 框架下的海量数据 Skyline 查询.

前置 Skyline 查询算法的基本原理是：在一部分 Map 任务完成后，Master 节点可以运用上述任何一种 Skyline 查询算法产生一个当前的全局过滤值，其余没有运行的 Map 任务可以在其初始化阶段从 Master 节点获取一个全局过滤值对自身的数据进行过滤，过滤掉一部分不能成为 Skyline 查询结果的数据，并将过滤后的数据作为输入，这样可以大大减少 Map 任务的输入量. 当系统中的 Map 任务数量很多的时候，前置 Skyline 查询算法可以看作是对贪婪 Skyline 查询算法和混合 Skyline 查询算法的优化，能够更有效地提高 Skyline 查询的效率.

图 8 仍以前述的 Skyline 查询为例，说明前置 Skyline 查询算法的基本过程. 原始数据集同样被分成 5 个分片，由 5 个 Map 任务完成，本文用 5 个步骤对前置 Skyline 算法进行简要说明.

步 1. 第 1 和第 2 个 Map 任务完成后，分别得到其局部过滤值，其中第 1 个 Map 任务的 Skyline 结果为点(3,3)和(4,2)，局部过滤值为点(3,3)，第 2 个 Map 任务的 Skyline 结果为点(5,5)和(7,2)，局部过滤值为点(5,5)，第 1 和第 2 个 Map 任务将点(3,3)和(5,5)发送给 Master 节点；

步 2. Master 节点在接收到点(3,3)和点(5,5)后，选择点(3,3)作为全局过滤值并发送给第 1 和第 2 个 Map 任务；

步 3. 第 1 和第 2 个 Map 任务将运用全局过滤值点(3,3)过滤后的 Skyline 查询结果发送给 Reduce 任务；

步 4. 第 3 到 5 个 Map 任务在启动之前，初始化阶段均从 Master 节点获取当前全局过滤值即点(3,3)来过滤自身原始数据集，其中点(3,3)可以“支配”第 3 个 Map 任务的所有原始数据，说明这些原始数据一定不是最终 Skyline 查询

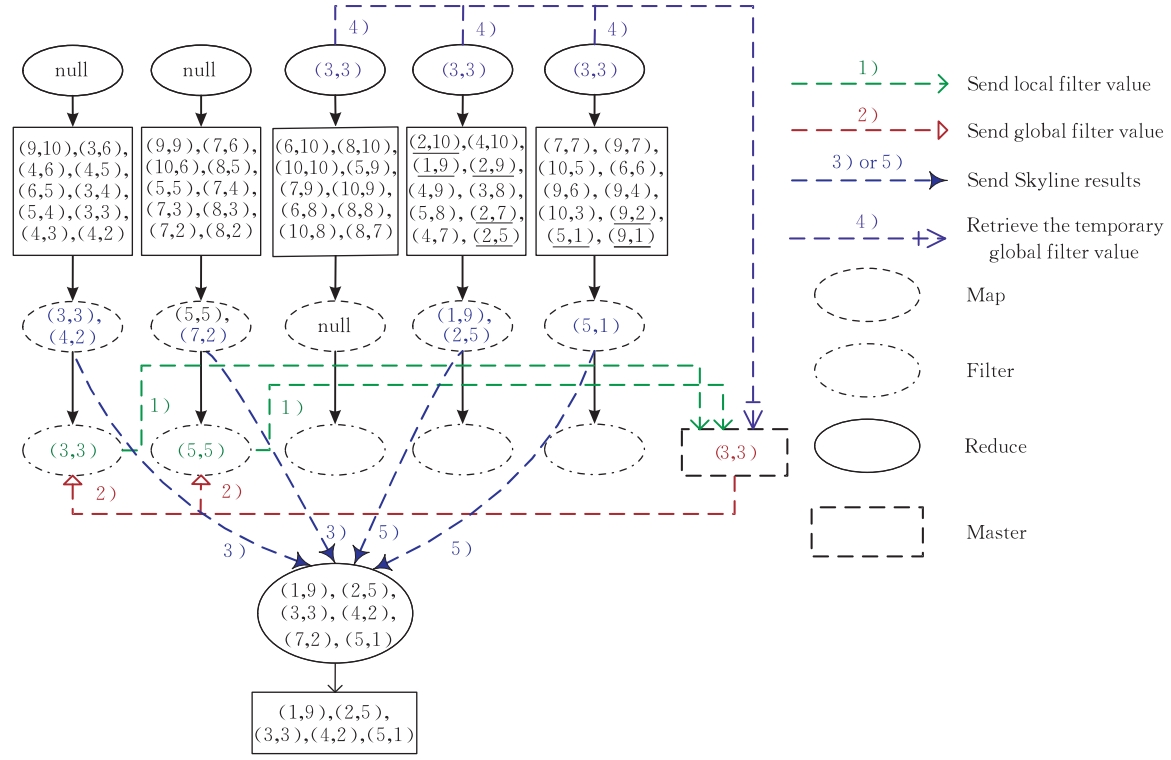


图 8 前置 Skyline 算法

结果，第 3 个 Map 任务的输入数据集为空集，即不会产生任何输出. 对于第 4 个 Map 任务，点(3,3)可以“支配”其点(4,10)，(4,9)，(3,8)，(5,8)，(4,7)，而不能“支配”其点(2,10)，(1,9)，(2,9)，(2,7)和(2,5)，即用下划线表示的点，则第 4 个 Map 任务的输入经过过滤后由 10 个点减少到 5 个点. 同理，第 5 个 Map 任务的输入经过过滤后降低到 3 个点，点(9,2)，(5,1)和(9,1)；

步 5. 后续 Map 任务将自身的 Skyline 查询结果发送给 Reduce 任务，得到最终 Skyline 查询结果.

由此可见，前置 Skyline 算法能够降低 Map 任务的输入量，更有效地提高了算法的效率.

5　实验评估

5.1　实验环境

本文的实验环境由 9 台用高速千兆网络连接的 PC 机组成，每个 PC 机的配置为 Quad Core 2.66GHZ CPU，4GB 内存，操作系统为 CentOS Linux 5.6. 其中一台 PC 机作为 Master 节点，其它 8 台 PC 机作为 Slave 节点. 本文采用的 Hadoop 平台版本为 0.20.2，代码编译采用 JDK1.6.

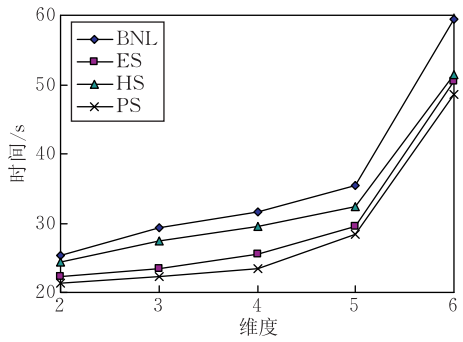
实验所用数据集为合成数据集，利用文献[1]中标准的数据生成工具生成. 实验数据集的记录条数为 2×10^5 条到 10×10^5 条，默认维度是 4，节点数为 8. 通常情况下测试 Skyline 查询性能在 3 种分布下测试，分别是独立分布、相关分布和反相关分布. 独立分布即数据的各个维度之间互相独立且同分布，相关分布是指数据在某一维的数值越大在其它维度上数值也越大，反相关分布是指数据在某一维上的值越大在其它维度上的值却越小. 由于数据在独立分布和相关分布下，Skyline 查询的性能相似，故本文并未对数据在相关分布下的性能进行测试，只选择了独立分布和反相关分布条件下测试 Skyline 查询算法的性能.

本文所采用的性能测试指标主要有两个：Skyline 查询算法的运行时间和查询记录数. 实验中，本文将所提出的 Skyline 查询算法，即贪婪 Skyline 查询算法(Eager Skyline Query Algorithm，ES)、混合 Skyline 查询算法(Hybrid Skyline Query Algorithm，HS)和前置 Skyline 查询算法(Prepositive Skyline Query Algorithm，PS)，与文献[13-14]的基于 Map-Reduce 框架的块嵌套循环算法(MR-BNL)进行比较. 由于延迟 Skyline 查询算法需要等待所有的 Map 任务运行结束才能产生全局过滤值，这种方法在实际应用中是不可行的，故本文并未对延迟 Skyline 查询算法进行测试. 本文将在 Map-Reduce 框架下 Reduce 任务的输入记录条数简写成(Reduce Input，RI)，则 BNL 算法的 Reduce 任务的输入记录数简写成(RI-BNL)，贪婪 Skyline 查询算法的 Reduce 任务输入记录数简写成(RI-ES)，混合 Skyline 查询算法的 Reduce 任务输入记录数简写成(RI-HS)，前置 Skyline 查询算法的 Reduce 任务输入记录数简写成(RI-PS).

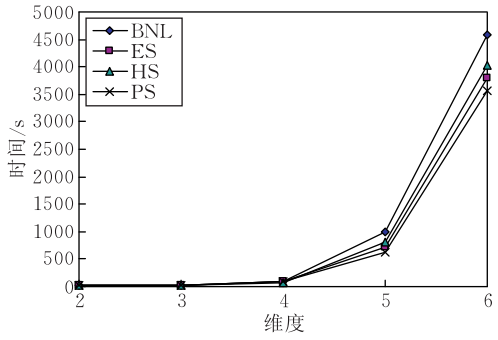
5.2　实验结果与分析

5.2.1　Skyline 查询性能随数据维度变化情况

图 9 和图 10 表示 Skyline 查询性能随数据维度变化的情况. 图 9 所示为 Skyline 查询算法运行时间的测试结果，图 10 所示为 Skyline 查询算法 Reduce 任务输入记录数的测试结果. 其中图 9(a)和 10(a)表示数据服从独立分布，图 9(b)和 10(b)表示数据服从反相关分布. 从图 9 可以看出，4 个 Skyline 算法的时间相差不大，但 BNL 算法的时间仍高于本文的 3 个算法，原因是本文的实验环境是在一个千兆网络环境下进行的测试，网络的传输时间很小，故时间性能的效果并不明显. 反相关分布的 Skyline 查询算法的时间在数据达到 5 维时时间明显增加，原因是反相关分布的 Skyline 查询结果比独立分布多，计算量增大，导致时间增加. 在图 10 中两种分布



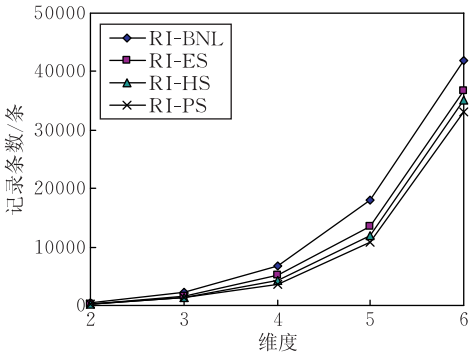
(a) 独立分布



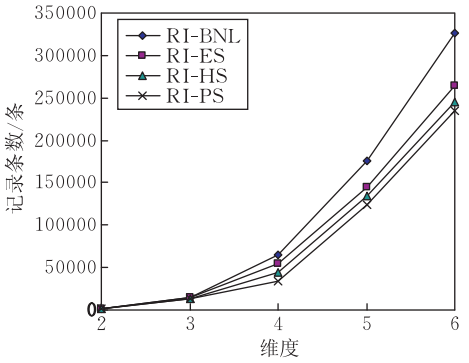
(b) 反相关分布

图 9　Skyline 查询算法运行时间

下 BNL 算法的 Reduce 任务输入记录数均高于本文的 ES、HS 和 PS 算法，表明本文所提出的 Skyline 查询算法可以减少 Reduce 任务输入记录数，即算法具有高效的过滤能力，可以提高 Skyline 算法的效率.



(a) 独立分布



(b) 反相关分布

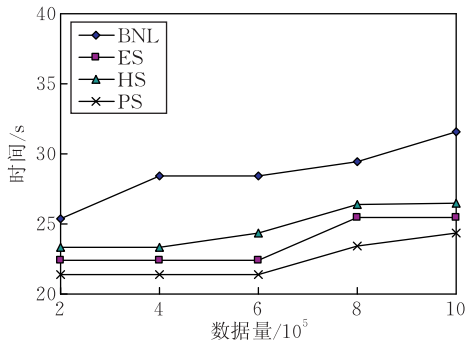
图 10 Skyline 查询算法 Reduce 任务输入记录数

5.2.2 Skyline 查询性能随数据量变化情况

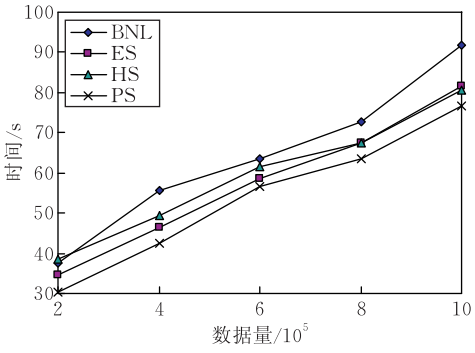
图 11 和图 12 表示 Skyline 算法性能随数据量变化的情况，可以看出随着数据量增大，Skyline 查询时间增加，原因是随着数据量增大，每个 Map 任务所处理的数据量也增大，使得算法的开销增大，本文的数据量从 2×10^5 条到 10×10^5 条，能够说明算法对于海量数据处理的性能. 从图 11 可以得出，在两种分布下，BNL 算法的运行时间高于本文的 3 个 Skyline 算法，而 PS 算法的性能最优，原因是 PS 算法在 Map 任务运行前过滤掉部分不能成为 Skyline 查询结果的数据，使得 Map 任务的输入量减少，故运行时间最短，同样从图 12 所示的 Reduce 任务输入记录数也能得到类似的结果.

5.2.3 Skyline 查询性能随节点数量变化情况

图 13 和图 14 表示 Skyline 查询算法性能随节点数变化的情况. 从图 13 中可以看出，随着节点数量的增加，Skyline 查询时间呈下降趋势，原因是伴随节点数量增加，系统的并行性和处理能力均增强，

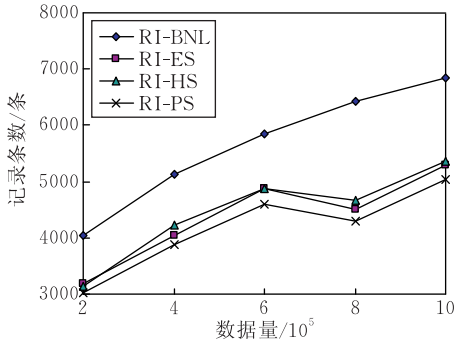


(a) 独立分布

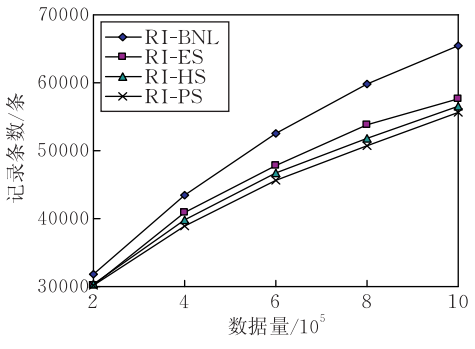


(b) 反相关分布

图 11 Skyline 查询算法运行时间



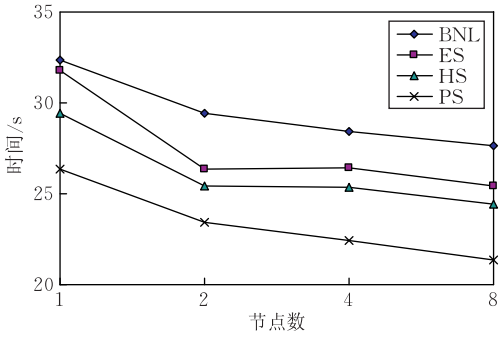
(a) 独立分布



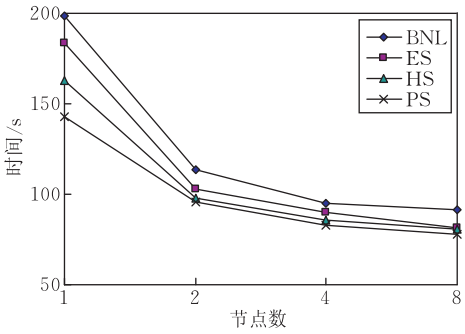
(b) 反相关分布

图 12 Skyline 查询算法 Reduce 任务输入记录数

故其时间也越来越短，而独立分布的运行时间小于反相关分布. 从图 14 中可以看出，在数据量一定的情况下，BNL 算法的 Reduce 任务的输入记录数保持

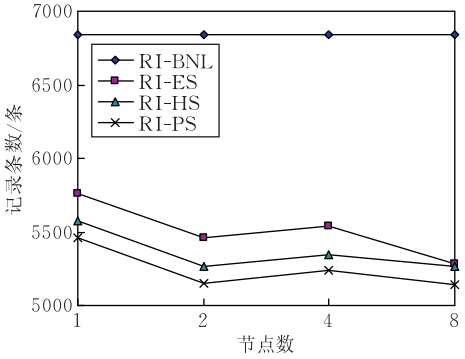


(a) 独立分布

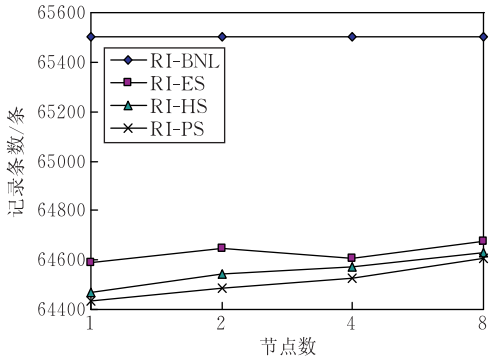


(b) 反相关分布

图 13　Skyline 查询算法运行时间



(a) 独立分布



(b) 反相关分布

图 14　Skyline 查询算法 Reduce 任务输入记录数

不变，原因是 BNL 算法并未采用过滤机制，故其 Reduce 任务的输入记录数保持不变. 而本文所提出的 Skyline 查询算法无论是在独立分布还是在反相关分布下 Reduce 任务输入记录数均大幅度少于 BNL 算法，表明本文算法具有很好的过滤能力，有效地提高了在 Map-Reduce 框架下处理 Skyline 查询的性能.

6　结　论

本文针对在云计算的 Map-Reduce 框架下如何实现海量数据 Skyline 查询问题进行了深入的研究. 首先，本文分析了基本 Skyline 查询算法，即在 Map-Reduce 框架下实现 Skyline 查询，其次，针对海量数据 Skyline 查询的最终结果的数量远小于原始数据集的数据量的问题，本文提出了 3 个 Skyline 查询算法，分别是延迟 Skyline 查询算法、贪婪 Skyline 查询算法和混合 Skyline 查询算法，有效地过滤部分不能成为最终查询结果的数据对象，并详细地分析每个算法的原理和特点. 为了进一步提高在 Map-Reduce 框架下处理海量数据 Skyline 查询的性能，本文针对上述算法又提出了一个优化算法，称为前置 Skyline 查询算法，前置 Skyline 查询算法更适用于处理海量数据处理 Skyline 查询. 最后，本文通过大量合成数据对所提出的 Skyline 查询算法进行了测试和评估. 实验结果表明，本文所提出的算法有效地节约了 Skyline 查询的时间，提高了 Skyline 查询的效率，算法具有良好的可扩展性、准确性和可用性.

参　考　文　献

[1] Borzsonyi S, Kossmann D, Stocker K. The Skyline operator//Proceedings of the ICDE. Washington, DC, USA, 2001: 421-430

[2] Wei Xiao-Juan, Yang Jing, Li Cui-Ping et al. Skyline query processing. Journal of Software, 2008, 19(6): 1386-1400(in Chinese)
(魏小娟，杨婧，李翠平等. Skyline 查询处理. 软件学报，2008，19(6)：1386-1400)

[3] Dean J, Ghemawat S. MapReduce: Simplified data processing on large cluster. Communications of the ACM, 2005, 51(1): 107-113

[4] Tan K L, Eng P K, Ooi B C. Efficient progressive Skyline computation//Proceedings of the VLDB. Roma, Italy, 2001: 301-310

[5] Kossmann D, Ramsak F, Rost S. Shooting stars in the sky: An online algorithm for Skyline queries//Proceedings of the VLDB. Hong Kong, China, 2002: 275-286

[6] Zhou Hong-Fu, Gong Xue-Qing, Zhen Kai, Zhou Ao-Ying. CSky: An online efficient algorithm for subspace skyline computation in high dimensional space. Chinese Journal of

Computers, 2007, 30(8):1409-1417(in Chinese)
(周红福,官学庆,郑凯,周傲英. 基于高维空间的在线高效子空间 Skyline 算法-CSky. 计算机学报, 2007, 30(8): 1409-1417)

[7] Wolf-Tilo Balke, Ulrich Güntzer, Jason Xin Zheng. Efficient distributed Skylining for web information systems//Proceedings of the EDBT. Heraklion, Crete, Greece, 2004: 256-273

[8] Wu Ping, Zhang Cai-Jie, Feng Ying et al. Parallelizing skyline queries for scalable distribution//Proceedings of the EDBT. Munich, Germany, 2006: 112-130

[9] Xin Jun-Chang, Wang Guo-Ren, Chen Lei et al. Continuously maintaining sliding window Skylines in a sensor network//Proceedings of the DASFAA. Bangkok Thailand, 2007: 509-521

[10] Dittrich J, Quiane-Ruiz J-A, Jindal A et al. Hadoop++: Making a yellow elephant run like a cheetah(without it even noticing). Proceedings of the VLDB Endowment, 2010, 3(1): 518-529

[11] Bu Y, Howe B, Balazinska M et al. HaLoop: Efficient iterative data processing on large clusters. Proceedings of the VLDB Endowment, 2010, 3(1): 285-296

[12] Yang Hung-Chih, Dasdan Ali, Hsiao Ruey-Lung et al. Map-reduce-merge: Simplified relational data processing on large clusters//Proceedings of the SIGMOD. Beijing, China, 2007: 1029-1040

[13] Zhang Bo-Liang, Zhou Shui-Geng, Guan Ji-Hong. Adapting Skyline computation to the MapReduce framework: Algorithms and experiments//Proceedings of the DASFAA Workshops. Hong Kong, China, 2011: 403-414

[14] Zhang Bo-Liang, Zhou Shui-Geng, Guan Ji-Hong. Skyline computation under MapReduce framework. Journal of Frontiers of Computer Science & Technology, 2011, 5(5): 385-397(in Chinese)
(张波良,周水庚,关佶红. MapReduce 框架下的 Skyline 计算. 计算机科学与探索,2011, 5(5): 385-397)

DING Lin-Lin, born in 1983, Ph.D.. Her research interests include cloud data management and P2P data management.

XIN Jun-Chang, born in 1977, Ph. D., lecturer. His research interests include sensor data management and uncertain data management.

WANG Guo-Ren, born in 1966, professor, Ph. D. supervisor. His research interests include XML data management, bioinformatics, distributed database and parallel computing, multimedia index technology, etc.

HUANG Shan, born in 1986, Ph. D. candidate. His research interests focus on cloud data management.

Background

Recently, Skyline query has been a research hot of Database and Information Retrieval. It is widely used in multi-objective decision making and data visualization. With the rapid increase of the amount of data for collecting and using by human, how to process Skyline query of massive data is becoming an urgent problem. The existing research on it is very rare. Although recent Skyline query algorithms are suit for centralized and distributed environment, the query performance reduces when processing Skyline query of massive data. Map-Reduce is a new parallel programming model that processes vast number of data on large clusters with easy deployment. Therefore, Map-Reduce can be used to solve Skyline query. In this paper, we aim at resolving the problem of processing Skyline query of massive data by Map-Reduce framework.

A straightforward implementation of Skyline query on Map-Reduce needs to scan all the candidate results before obtaining the final results. However, when the amount of final results is much smaller than the original data, there is a waste of processing unnecessary results on Map-Reduce. In this paper, a series efficient Skyline query algorithms, Lazy Skyline Query Algorithm, Eager Skyline Query Algorithm, Hybrid Skyline Query Algorithm, have been proposed to prune the unpromising results effectively and enhance the performance of query processing effectively. The main idea of the three algorithms is that we prune the output of Mappers by generating global filter values, so as to decrease the input of Reducer. The principle of Lazy Skyline Query Algorithm is that the Master node generates a global filter value after all Mappers completing. The main principle of Eager Skyline Query Algorithm is that after one Mapper completing, the Master node generates a filter value used to prune the output of this Mapper. Hybrid Skyline Query Algorithm is a combination of the above two algorithms. The Master node waits for a time interval and then generates a filter value. In addition, we propose an optimization, Prepositive Skyline Query Algorithm, to prune the input of Mappers in their initial phase and enhance the performance of processing Skyline query dramatically. Our Skyline query algorithms are efficient, scalable and available.

This work is supported by Major Program of National Natural Science Foundation of China under grant of No. 60933001, National Natural Science Funds for Distinguished Young Scholar under grant of No. 61025007 and the Fundamental Research Funds for the Central Universities under grant of No. N090304007. Our research achievements can enhance the performance of processing Skyline query of massive data.