基于 MapReduce 的 Skyline 查询处理算法

崔文相[1,2] 肖迎元[1,2] 郝 刚[1,2] 王洪亚[3] 邓华锋[4]
(天津理工大学计算机与通信工程学院 天津 300384)[1]
(天津市智能计算及软件新技术重点实验室 天津 300384)[2]
(东华大学计算机科学与技术学院 上海 201620)[3] (江西师范大学计算机信息工程学院 南昌 330022)[4]

摘 要 Skyline 查询是一个典型的多目标优化查询,在多目标优化、数据挖掘等领域有着广泛的应用。现有的 Skyline 查询处理算法大都假定数据集存放在单一数据库服务器中,查询处理算法通常也被设计成针对单一服务器的串行算法。随着数据量的急剧增长,特别是在大数据背景下,传统的基于单机的串行 Skyline 算法已经远远不能满足用户的需求。基于流行的分布式并行编程框架 MapReduce,研究了适用于大数据集的并行 Skyline 查询算法。针对影响 MapReduce 计算的因素,对现有基于角度的划分策略进行了改进,提出了 Balanced Angular 划分策略;同时,为了减少 Reduce 过程的计算量,提出了在 Map 端预先进行数据过滤的策略。实验结果显示所提出的 Skyline 查询算法能显著提升系统性能。

关键词 MapReduce,Skyline,数据划分
中图法分类号 TP338.8 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.6.007

MapReduce-based Skyline Query Processing Algorithm

CUI Wen-xiang[1,2] XIAO Ying-yuan[1,2] HAO Gang[1,2] WANG Hong-ya[3] DENG Hua-feng[4]
(School of Computer and Communication Engineering, Tianjin University of Technology, Tianjin 300384, China)[1]
(Tianjin Key Laboratory of Intelligence Computing and Novel Software Technology, Tianjin 300384, China)[2]
(Department of Computer Science and Engineering, Donghua University, Shanghai 201620, China)[3]
(School of Computer and Information Engineering, Jiangxi Normal University, Nanchang 330022, China)[4]

Abstract Skyline query is a typical multi-objective optimization problem and is widely applied in multi-objective optimization, data mining and other fields. Most of the existing Skyline query processing algorithms assume that the data set is placed in a single server, and query processing algorithm is designed as a serial algorithm for a single server. With the rapid growth of data, especially under the background of big data, the traditional serial Skyline algorithms based on a single computer are far from enough to meet the needs of users. Based on the popular distributed parallel programming framework MapReduce, this paper studied the parallel skyline query algorithm suitable for large data sets. Aiming at the factors affecting MapReduce, this paper improved the existing data partition strategy based on angles and proposed the data partition strategy based on Balanced Angular. Meanwhile, to reduce the computation of Reduce phase, this paper proposed the data filtering strategy in advance at Map. The experimental results show that the proposed Skyline query algorithm can improve system performance significantly.

Keywords MapReduce, Skyline, Data partition

1 引言

Skyline 查询是一个典型的多目标优化问题[1,2],它起源于极大向量问题[3,4]。2001 年 Börzsönyi 将其扩展为一种基本查询操作并引入到数据库系统中,以支持对存储于磁盘的大量多维数据的"多目标最优"查询[5]。由于在多目标优化、数据挖掘等领域有着重要的应用,Skyline 查询问题一直受到学术界和工业界的极大关注和重视。

具体来讲,Skyline 查询是指从一个给定的多维数据对象集合 S 中挑选出不被 S 中任何数据对象支配(dominate)的所有数据对象。我们说数据对象 u 支配数据对象 q,是指 u 在所有维上都不比 q 差,并且至少在某一维上比 q 好。

现有的 Skyline 查询处理算法大都假定数据集存放在单一数据库服务器中,查询处理算法通常也被设计成针对单一服务器的串行算法,如 BNL(块嵌套循环算法)[5]、D&C(分而治之算法)[5]、SFS(排序优先 Skyline 算法)[6]、LESS(线性消

到稿日期:2015-06-25 返修日期:2015-09-02 本文受国家自然科学基金(61170174,61370205),天津市创新团队计划(TD12-5016)资助。
崔文相(1990—),男,硕士生,主要研究方向为云计算、Skyline 查询等;肖迎元(1969—),男,博士,教授,博士生导师,主要研究方向为数据库、移动计算等,E-mail:yyxiao@gmail.com;郝 刚(1968—),男,博士,主要研究方向为数据库、智能计算等;王洪亚(1976—),男,博士,教授,博士生导师,主要研究方向为数据库、实时系统等;邓华锋(1974—),男,博士,副教授,博士生导师,主要研究方向为数据库、智能计算等。

除排序 Skyline 算法)[7]、SaLSa (排序限定 Skyline 算法)[8]、Bitmap(位图算法)[9]、BBS(分支界定算法)[10,11]等经典 Skyline 算法都是基于单服务器的串行算法。在数据量不大的情况下,上述算法都表现出了很好的性能。然而,随着数据量的急剧增长,特别是在大数据背景下,传统的基于单服务器的串行 Skyline 算法已经远远不能满足用户的需求。因此,研究支持海量数据集的分布式并行 Skyline 查询算法已成为 Skyline 研究的一个新方向。

MapReduce 是由 Google 提出的流行的分布式并行编程框架,用于大规模数据集的并行计算[12]。基于 MapReduce 来设计支持海量数据集的分布式并行 Skyline 查询算法成为一种自然的选择。Liang 等对现有的串行 Skyline 算法 BNL、SFS 和 Bitmap 进行改进,实现了基于 MapReduce 编程框架的相应并行算法(即 MR-BNL、MR-SFS 和 MR-Bitmap)[13],并在 MapReduce 的开源平台 Hadoop 上加以实现,但上述基于 MapReduce 的并行 Skyline 算法并没有针对 MapReduce 的特点做相应的优化。Chen 等针对 MapReduce 提出了一种基于角度的数据集划分方法来优化 Skyline 计算[14],但是却忽略了 MapReduce 框架下影响集群处理速度的两个重要因素:1)Map 与 Reduce 之间的传输量;2)分配到各个计算节点的计算量。如果不考虑 Map 与 Reduce 之间的传输量,算法将受到传输带宽的限制,并且如果中间结果较大,Map 需要将中间计算结果写入磁盘,从而带来了大量的磁盘 I/O。负载均衡问题是 MapReduce 计算中经常遇到的问题,如果任务划分得不合理,会导致整个计算过程受制于单个任务而造成瓶颈;并且对于拥有大量计算的任务来说,一旦任务失败必须重做,更加影响计算进度。Hadoop 可以通过设置任务重做次数来缓解这个问题,但也只能在预测到单个任务有可能失败时重新启动一个任务来计算,并不能真正解决负载的均衡性问题。本文针对影响 MapReduce 计算的因素,对基于角度的划分策略进行了改进,提出了 Balanced Angular 划分策略(简称 BA);同时,为了减少 MapReduce 实施 BNL 算法时的 Combine 以及 Reduce 过程的计算量,提出了在 Map 端过滤数据的策略。

本文第 2 节介绍了 Skyline 查询的定义以及 MapReduce 计算框架的基本情况;第 3 节详细介绍了提出的基于 Balanced Angular 划分和过滤策略的并行 Skyline 算法;第 4 节对所提算法的性能进行了实验分析;最后总结全文。

2 背景知识

本节首先给出了 Skyline 查询的形式化定义,然后介绍了 MapReduce 计算框架。

2.1 Skyline 查询

在下面的描述中,用 S 表示给定的多维数据对象集合;用 $P=\{p_1,p_2,\cdots,p_k\}$ 代表 S 中数据对象的维度集合;对于任意的 $u\in S$,用 $u[p_i]$ 来表示数据对象 u 在第 i 个维度 p_i 上的值,这里 $1\leqslant i\leqslant k$。

定义 1(支配关系 $\prec$) 对任意的 $u,q\in S$,若($\forall p_i\in P$, $u[p_i]\leqslant q[p_i]$) 且($\exists p_j\in P, u[p_j]<q[p_j]$),则称 u 支配 q,简记为 $u\prec q$。

定义 2(支配力) 对任意的 $u\in S$,称 u 可以支配的 S 中的数据对象的个数 nu_dom 为 u 的支配力。

定义 3(Skyline 查询) 对于一个给定的多维数据对象集合 S,Skyline 查询返回 S 中所有不被任何其它对象支配的数据对象的集合,本文用 $sk(S)$ 表示 S 上 Skyline 查询的结果集,即 $sk(S)=\{\forall u\in sk(S)\mid$不存在 $o\in S$ 使得 $o\prec u\}$。若 $u\in sk(S)$,则称 u 为 S 上的 Skyline 点。

定义 4(数据集的划分) 假定 $S_1,S_2,\cdots,S_m$ 都是数据集 S 的子集,若 $S_1\cap S_2\cap\cdots\cap S_m$ 为空并且 $S_1\cup S_2\cup\cdots\cup S_m=S$,则称 $S_1,S_2,\cdots,S_m$ 为数据集 S 的一个划分。

若 $S_1,S_2,\cdots,S_m$ 是数据集 S 的一个划分;$sk(S_1)$, $sk(S_2),\cdots,sk(S_m)$ 分别表示在 $S_1,S_2,\cdots,S_m$ 上的 Skyline 查询的结果集;$S_{1,2,\cdots,m}=sk(S_1)\cup sk(S_2)\cup\cdots\cup sk(S_m)$。容易证明 Skyline 计算具有如下性质:

性质 1 $sk(S_{1,2,\cdots,m})=sk(S)$

性质 1 说明了在一个数据集 S 上执行 Skyline 查询可以转化为:首先对 S 的一个划分的各子集分别执行 Skyline 查询,然后再对它们的结果集的并集执行 Skyline 查询。例如,为了计算集合 S 的 Skyline 查询结果集 $sk(S)$,可以首先找到 S 的一个划分:$S_1,S_2,\cdots,S_m$,然后对 $S_1,S_2,\cdots,S_m$ 分别计算它们的 $sk(S_1),sk(S_2),\cdots,sk(S_m)$,最后再对它们的并集 $sk(S_1)\cup sk(S_2)\cup\cdots\cup sk(S_m)$ 执行一次 Skyline 查询,得到的结果集等价于 $sk(S)$。

上述性质为我们采用 MapReduce 编程框架设计并行 Skyline 查询算法提供了依据,即首先找到 S 的一个划分 S_1, $S_2,\cdots,S_m$,然后针对各数据分片分别计算局部 Skyline 查询,得到 $sk(S_1),sk(S_2),\cdots,sk(S_m)$,最后将它们发送给 Reduce,Reduce 对汇聚后的结果集再执行一次 Skyline 查询,得到最终结果集。

2.2 MapReduce 计算框架

MapReduce 作为流行的分布式并行编程框架,得到了学术界和工业界的高度重视;Hadoop 作为其开源实现,目前已成为大数据处理的主流平台。MapReduce 计算框架原理如图 1 所示,一个 MapReduce 任务通常被分解为 Map 和 Reduce 两阶段执行,任务开始时作业调度器为每一个 Hadoop 分布式文件系统(HDFS)中的数据分片启动一个 Map 任务:将原数据以〈Key1,Value1〉的键值对形式输入,运算产生中间结果〈Key2,List(Values)〉。接着进行 Combine 运算,该过程将同一分片内具有相同 Key 值的 Value 值予以合并,然后将结果传送给 Reduce 端,Reduce 将根据各 Key 值计算出的最终结果予以输出。Combine 通常会使用 Reduce 实例进行初始化,这可以大大减少计算框架内数据的传输量,从而提升计算框架的效率。

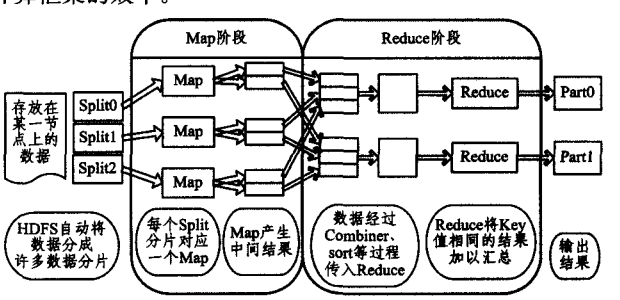

图 1 MapReduce 计算框架

3 基于 Balanced Angular 划分的并行 Skyline 算法

3.1 算法基本思想

本文提出的基于 Balanced Angular 划分的并行 Skyline 算法(简记为 BAPS)以单机串行 BNL 算法为原型,针对 MapReduce计算框架的特点将 Skyline 查询划分成 3 个阶段:预处理阶段、计算阶段和整合阶段。预处理阶段通过分析数据的特征计算出合适的数据划分策略(Balanced Angular 划分),从而确保计算阶段各节点计算的负载均衡;在计算阶段使用 Map 任务对数据集进行过滤,以减少 Combine 以及 Reduce 的计算量。计算阶段的 Reduce 计算出拥有相同 Key 值的分区内的局部 Skyline 点;整合阶段汇集计算阶段的各局部 Skyline 点,最终得到 Skyline 查询的结果集,即 $sk(S)$。

3.2 预处理阶段

预处理阶段的核心是生成数据划分策略(Balanced Angular 划分)。Balanced Angular 划分以文献[13]提出的基于角度的划分策略为基础,针对该划分没有考虑并行计算中负载均衡的问题进行了改进。以一个二维数据集为例,基于角度划分的策略如图 2 所示,该方式将数据集按角度均匀划分成了 4 个子区域。这种划分策略过于机械,很容易造成计算量分配不合理。图 3 显示了基于角度的划分策略针对某一数据集出现的问题:按照基于角度的划分策略划分出的 1 号区域没有任何数据,浪费了 Hadoop 集群计算量,而 3 号区域又因要处理的数据量过多而增加了整个任务的计算时间。

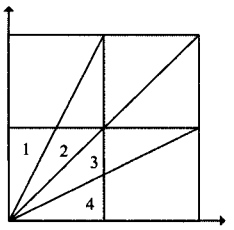

图 2 基于角度的划分策略

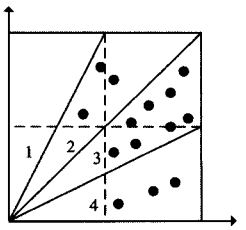

图 3 角度划分不合理

本文提出的 Balanced Angular 的划分策略是通过计算数据集的弧度平均值而得到。Balanced Angular 在划分时首先确定需要划分的数据块数目,在独立同分布的数据集中 Skyline 点的数目为 $sn=\Theta((\ln n)^{k-1}/(k-1)!)$[9],其中 n 为数据的总个数,k 为数据的维度。由计算式可以看出 Skyline 点的数目与进行 Skyline 查询的维度的关系最强。为此,本文设置数据集分区数目为 $n_{div}=2^{k-1}$(k 为 Skyline 查询的维度)。具体来说,Balanced Angular 在数据集的每个球面坐标维度的弧度均值处进行划分。球面坐标的计算方式如下。

假设 $u=(u[p_1],u[p_2],\cdots,u[p_k])$ 为多维数据集 S 中的某一数据对象,其中 $u[p_1],u[p_2],\cdots,u[p_k]$ 分别代表了数据对象 u 在 k 个维度上的坐标值,将该数据对象的直角坐标转化到球面坐标为 $(\tan(\varphi_1),\tan(\varphi_2),\cdots,\tan(\varphi_{k-1}))$,其中 $\tan(\varphi_1)=\frac{\sqrt{u[p_k]^2+u[p_{k-1}]^2+\cdots+u[p_2]^2}}{u[p_1]},\cdots,\tan(\varphi_{k-1})=\frac{\sqrt{u[p_k]^2}}{u[p_{k-1}]}$。可以看出,将数据对象从直角坐标转化到球面坐标后原来的 k 维点被降维成 $k-1$ 维点;在此基础上,可以计算出该数据对象在球面坐标的第 i 维的弧度值,其中 $1\leqslant i\leqslant k-1$;进一步,可以计算出数据集 S 上各点在球面坐标各维度上的弧度均值:$(\overline{\varphi_1},\overline{\varphi_2},\cdots,\overline{\varphi_{k-1}})$,并以此作为划分依据。

图 4 示出采用本文提出的 Balanced Angular 划分策略在二维数据集上的划分效果。由于二维数据点变换为球面坐标后为一维值,Balanced Angular 划分策略选择数据集中各点在该维度上的弧度均值作为划分角度。从图 4 可以看出数据集被划分成了两个相对均匀的区域,解决了基于角度划分的数据集分布不均的问题。

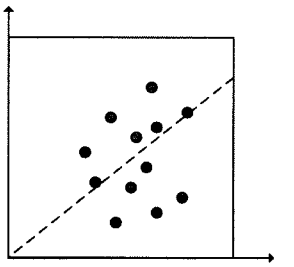

图 4 Balanced Angular 划分策略

3.3 计算阶段

为了减少 Skyline 查询处理过程的计算量,在 Map 阶段从局部 Skyline 点中选出支配能力最大的前 j 个作为过滤值,以减小后续 Combiner 以及 Reduce 阶段的数据计算量,从而加快算法的处理速度。Skyline 计算的复杂度为 $O(kn^2)$,其中 k 表示数据的维度,n 为数据个数。过滤的计算代价为 $O(kn)$。从理论上讲过滤的时间代价可以弥补后续 Skyline 点计算的代价。但过滤的性能主要由选出的过滤值的支配力来决定。在之后的实验中会给出相应的详细分析。

在进行 MapReduce 任务设置时,Reduce 的数量一般有两种设置原则:1)设置为 $0.95*(n_{node}*n_{core})$;2)设置为 $1.75*(n_{node}*n_{core})$,这里 n_{node} 指的是节点的个数,n_{core} 代表每个节点的核心数。对于第一种设置,可以同时启动所有 Reduce 任务;对于第二种设置,可以在第一轮 Reduce 完成后马上开始第二轮 Reduce 任务。Reduce 数目设置过多容易加重合并阶段的计算负担,本文设置 Reduce 的数目为 $n_{reduce}=\min\{0.95*n_{node}*n_{core},2k\}$,这样既兼顾了负载均衡,又能减轻计算负担。

该阶段的 Map 任务可以描述为:

Map()

算法输入:格式化数据〈Key,Value〉

算法输出:键值对〈Key,ValueList vl〉

1. 在分片内计算前 n_r 个数据中支配力最大的 i 个数据点作为过滤值;
2. 在后续数据局部 Skyline 查询计算中使用过滤值判断是否可直接过滤;
3. 计算数据所属的划分区域作为 Key 值;
4. 输出:〈Key,ValueList vl〉;

Reduce 任务与 Combiner 的功能相同,其处理过程描述如下:

Reduce()/Combiner ()

算法输入:各 Map 任务的输出〈Key,ValueList vl〉

算法输出:根据各 Key 值划分的逻辑分区内的 Skyline 点集

1. 输入:〈Key,ValueList vl〉;
2. 计算 Key 值相同的区域内的 Skyline 点集 $sk(S_i)$;
3. 输出:〈Key,$sk(S_i)$〉。

该阶段完成后,会将由 Key 分割的各个逻辑分块内的局部 Skyline 点再进行一次结果集汇总,得到最终数据集的 Skyline 结果集。与计算过程不同,汇总阶段只能设置一个

Reduce 任务以保证计算出来的结果是针对全局的 Skyline。汇总阶段不再使用过滤以及分区机制，仅将前一阶段的输出作为输入，使用 BNL 算法进行一次 Skyline 计算。另外，汇总阶段可以采用由 Hadoop 的 API 所支持的多任务提交方式与计算阶段一起提交任务：该任务可以通过控制 job. waitForCompletion()为 true 的方式保证正式计算阶段的任务完成后才开始执行本阶段的任务，避免了多次提交任务的麻烦。

4 实验及分析

4.1 实验设置

采用了 9 节点的 Hadoop 平台，每台机器内存 15GB，处理器为 Intel(R) Xeon(R) E5440 @ 2.83GHz，操作系统为 64 位Red Hat Enterprise Linux Server release 6.2，JDK 版本为 1.6.0_22，Hadoop 版本为 2.2.0，HDFS 格式化时设置的数据分片大小为 128MB，JVM 内存最大为 1GB，设置Map-Reduce的 JVM 重用个数为 8。实验时采用线下打包上传至主节点运行 jar 包的方式进行测试，全部数据是上传到 HDFS 上测试的。

本实验采用了与文献[5]类似的数据集，为了节省存储空间并充分测试 Hadoop 框架的计算能力，舍弃了字节填充的策略。为了探究算法性能，设计了以下几种数据集。

(1)独立同分布数据集：各维度数据符合均匀分布且各维度独立同分布。

(2)独立非均匀数据集：各维度独立但是维度上各自拥有自己的特性。

(3)相关数据集：数据在一个维度上占优的一般在其他维度也占优。

(4)反相关数据集：数据一般在一个维度上占优但在其他维度上表现较弱。

图 5 以二维数据为例模拟显示了各类数据集的大致分布情况。

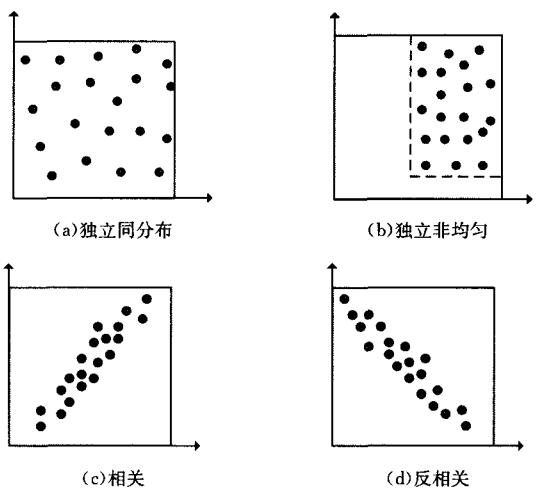

图 5 数据分布特点

各个维度的每类数据集都取相同数目的数据，数据取值区间为 0.00～100.99，保留小数点后两位。每类数据集的维度取值分别为 3，5，8，10，大小分别达到 1.7GB，2.8GB，4.4GB和 5.5GB。

4.2 实验结果

实验分为两个方面进行：首先验证新的划分策略的效果；然后测试在 Map 任务中加入过滤策略的效果。

4.2.1 数据划分策略

图 6 显示了本文提出的 Balanced Angular 数据划分策略(图中用 new 代表)和文献[14]提出的角度划分方法(图中用 old 代表)的性能比较，其中(a)、(b)、(c)、(d) 4 个子图分别对应图 5 中的(a)、(b)、(c)、(d) 4 个数据集。从图 6 中可以看出本文提出的数据划分策略在 4 个数据集上都明显好于文献[14]的划分策略，特别是本文提出的算法在处理相关数据集时的性能尤为突出，这主要是因为划分子任务的合理性使得分配到各个 Reduce 任务中的部分能够一次装入内存进行处理。

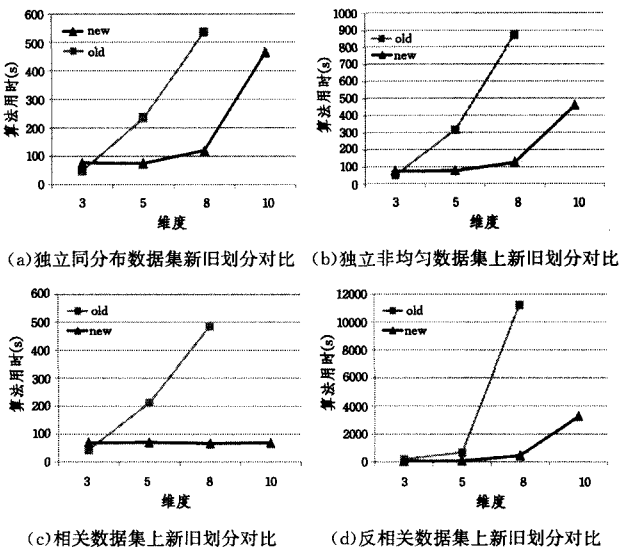

图 6 划分策略在不同数据集上的性能对比

此外，文献[14]提出的数据划分策略在数据维度高于 8 的数据集上没有性能数据，是因为基于该策略在 8 维以上数据集上执行效率过低而没能进行足够的实验。在计算低维数据集上的 Skyline 时，该策略因为要启动多个 Reduce 任务而耗费更多的时间，性能较原策略略有下降。

4.2.2 过滤策略

过滤策略有两个关键的测试点：Map 节点计算过滤点时所使用的数据点的数量(即 Map 算法描述中的 n_r)以及挑选的过滤点的个数(即 Map 算法描述中的 i)。两者增大都会带来计算量的增加并对过滤效果产生影响，下面实验着重考察 n_r 和 i 的取值如何影响过滤策略的性能。

下面的实验采用了独立同分布的 8 维数据集。实验中数据集的分片大小为 128M，每个分片大概包括 330 多万个数据。在确保 i 值固定的情形下测试算法性能随 n_r 变化的情况，n_r 的取值范围为 1000～5000。

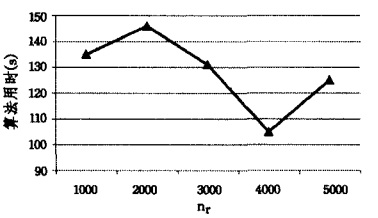

图 7 n_r 对过滤效果的影响

从图 7 可以看出 n_r 对算法性能的影响，当 n_r 的取值在

(下转第 64 页)

[J]. 计算机工程与应用,2009(27):203-205

[5] Tao Ping, Zhang Xiao-ying, Ma Heng-Liang. Simulation of personnel evacuation based on cellular automaton model[J]. Computer Simulation, 2009, 26(10): 319-322(in Chinese)
陶平,张小英,马恒亮. 基于元胞自动机模型的人员疏散仿真研究[J]. 计算机仿真,2009(10):319-322

[6] Murakami Y, Minami K, Kawasoe T, et al. Multi-agent simulation for crisis management[C]// Proceedings IEEE Workshop on Knowledge Media Networking. Kyoto, Japan: IEEE Computer Society, 2002: 135-139

[7] Hughes R L. The flow of human crowds[J]. Annual Review of Fluid Mechanics, 2003, 35: 169-182

[8] Kountouriotis V, Thomopoulos S C A, Papelis Y. An agent-based crowd behaviour model for real time crowd behaviour simulation[J]. Pattern Recognition Letters, 2014, 44(1): 30-38

[9] Xu Gao. Simulation model for crowd evacuation based on agent technology[J]. Journal of Southwest Jiaotong University, 2003, 38(3): 301-303(in Chinese)
徐高. 基于智能体技术的人员疏散仿真模型[J]. 西南交通大学学报,2003(3):301-303

[10] Cui Xi-hong, Li Qiang, Chen Jin, et al. Study on MA-based Model of Occupant Evacuation in Public Facility[J]. Journal of System Simulation, 2008, 20(4): 1006-1010(in Chinese)
崔喜红,李强,陈晋,等. 基于多智能体技术的公共场所人员疏散模型研究[J]. 系统仿真学报. 2008,20(4):1006-1010

[11] Huang Xi-fa, Wang Ke-jun, Guo Lian-ying, et al. Microscopic simulation model study on pedestrian evacuation based on agent technology[J]. Journal of System Simulation, 2009, 21(15): 4568-4571(in Chinese)
黄希发,王科俊,郭莲英,等. 基于 Agent 技术的人员疏散微观仿真模型研究[J]. 系统仿真学报,2009(15):4568-4571

[12] Hughes R L. A continuum theory for the flow of pedestrians [J]. Transportation Research Part B-Methodological, 2002, 36(6): 507-535

[13] Santra S B, Schwarzer S, Herrmann H. Fluid-induced particle-size segregation in sheared granular assemblies[J]. Physical Review E, 1996, 54(5): 5066-5072

[14] Makse H A, Havlin S, King P R, et al. Spontaneous stratification in granular mixtures[J]. Nature, 1997, 386(6623): 379-382

[15] Santra S B, Schwarzer S, Herrmann H. Fluid-induced particle-size segregation in sheared granular assemblies[J]. Physical Review E, 1996, 54(5): 5066-5072

(上接第 38 页)

1000～2000 之间时,算法用时随着 n_r 的增大而增多;当 n_r 的取值在 2000～4000 之间时,算法用时随着 n_r 的增大而减少;当 n_r=4000 时算法用时到达最小,此后算法用时又随着 n_r 的增大而增多。

进一步,在确保 n_r 值固定的情形下测试算法性能随 i 变化的情况,i 的取值范围为 1～5。图 8 显示了 i 对算法性能的影响,即随着 i 的增加,算法用时相应减少,这是因为随着过滤点个数的增加,能够预先过滤的数据对象也相应增加。

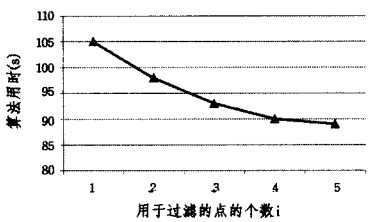

图 8 i 对过滤效果的影响

结束语 Skyline 查询在多目标优化、数据挖掘等领域有着重要的应用,传统的基于单机的串行 Skyline 查询算法在大数据量的情形下很难满足用户的需求。本文基于流行的分布式并行编程框架 MapReduce,提出了适用于在大数据集上计算 Skyline 查询的并行算法。该并行算法对现有的基于角度的划分策略进行了改进,提出了 Balanced Angular 划分策略和 Map 端过滤数据的策略。实验结果显示,本文提出的并行 Skyline 查询算法能显著提升系统性能。

参 考 文 献

[1] Balke W T, Güntzer U. Multi-objective query processing for database systems[C]// Int'l Conference on Very Large Data Bases (VLDB). 2004: 936-947

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

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

[4] Preparata F P, Shamos M I. Computational Geometry: An introduction[M]. New York: Springer-Verlag, 1985

[5] Börzsönyi S, Kossmann D, Stocker K. The skyline operator[C]// IEEE International Conference on Data Engineering (ICDE). 2001: 421-430

[6] Chomicki J, Godfrey P, Gryz J, et al. Skyline with presorting[C]// IEEE International Conference on Data Engineering (ICDE). 2003: 717-816

[7] Godfrey P, Shipley R, Gryz J, et al. Maximal vector computation in large data sets[C]// Int'l Conference on Very Large Data Bases (VLDB). 2005: 229-240

[8] Bartolini I, Ciaccia P, Patella M, et al. Efficient sort-based skyline evaluation[J]. ACM Transactions on Database Systems (TODS), 2008, 33(4): 1-45

[9] Tan K L, Eng P K. Efficient progressive skyline computation[C]// IEEE International Conference on Data Engineering (ICDE). 2001: 301-310

[10] Papadias D, Tao Y. An optimal and progressive algorithm for skyline queries[C]// ACM SIGMOD Int'l. Conference on Management of Data (SIGMOD). 2003: 467-478

[11] Papadias D, Tao Y. Progressive skyline computation in database systems[J]. ACM Transactions on Database Systems (TODS), 2005, 30(1): 41-82

[12] Dean J, Ghemawat S. MapReduce: Simplified Data Processing on Large Clusters[J]. Communications of the ACM, 2008, 51(1): 107-113

[13] Zhang Bo-liang, Zhou Shui-geng, Guan Ji-hong. Skyline under MapReduce framework[J]. Computer Science and Exploration, 2011, 5(5): 385-397(in Chinese)
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

[14] Chen L, Hwang K, Wu J. MapReduce Skyline Query Processing with A New Angular Partitioning Approach[C]// Proc. of International Parallel and Distributed Processing Symposium. 2012: 2262-2270