软件学报 ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*,2013,24(8):1836−1851 [doi: 10.3724/SP.J.1001.2013.04377]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

# MapReduce 框架下基于 R-树的 *k*-近邻连接算法*

刘 义, 景 宁, 陈 荦, 熊 伟

(国防科学技术大学 电子科学与工程学院,湖南 长沙 410073)

通讯作者: 刘义, E-mail: liu.yi.nudt@gmail.com

**摘 要**: 针对大规模空间数据的高性能*k*-近邻连接查询处理,研究了MapReduce框架下基于R-树索引的*k*-近邻连接查询处理.首先利用无依赖并行和串行同步计算的形式化定义抽象了MapReduce并行编程模型,基于此并行计算模型抽象,分别提出了R-树索引快速构建算法和基于R-树的并行*k*-近邻连接算法.在索引构建过程中,提出一种采样算法以快速确立空间划分函数,使得索引构建符合无依赖并行和串行同步计算抽象,在MapReduce框架下非常容易进行表达.在*k*-近邻连接查询过程中,基于构建的分布式R-树索引,引入*k*-近邻扩展框限定查询范围并进行数据划分,然后利用R-树索引进行*k*-近邻连接查询,提高了查询效率.从理论上分析了所提出算法的通信和计算代价.实验与分析结果表明,该算法在真实数据集的查询上具有良好的效率和可扩展性能,可以很好地支持大规模空间数据的*k*-近邻连接查询处理,具有良好的实用价值.

**关键词**: 云计算;MapReduce;*k*-近邻连接;空间查询;R-树

**中图法分类号**: TP311 **文献标识码**: A

中文引用格式: 刘义,景宁,陈荦,熊伟.MapReduce框架下基于R-树的*k*-近邻连接算法.软件学报,2013,24(8):1836−1851. http://www.jos.org.cn/1000-9825/4377.htm

英文引用格式: Liu Y, Jing N, Chen L, Xiong W. Algorithm for processing *k*-nearest join based on R-tree in MapReduce. Ruan Jian Xue Bao/Journal of Software, 2013,24(8):1836−1851 (in Chinese). http://www.jos.org.cn/1000-9825/4377.htm

## Algorithm for Processing *k*-Nearest Join Based on R-Tree in MapReduce

LIU Yi, JING Ning, CHEN Luo, XIONG Wei

(College of Electronic Science and Engineering, National University of Defense Technology, Changsha 410073, China)

Corresponding author: LIU Yi, E-mail: liu.yi.nudt@gmail.com

**Abstract**: To accelerate the *k*-nearest neighbor join (knnJ) query for large scale spatial data, the study presents a knnJ based on R-tree in MapReduce. First, the research uses the formalization of independent parallelism and sequential synchronization (IPSS) computation to abstract MapReduce parallel program model. Next, based on this parallel model abstraction, this paper proposes efficient algorithms for bulk building R-tree and performing knnJ query based on the constructed R-tree respectively. In the process of bulk building R-tree, a sampling algorithm is provided to determine the spatial partition function rapidly, which make the process of building R-tree conform to IPSS model and can be expressed easily in MapReduce. In the process of knnJ query, the knn expanded bounding box is introduced to limit the knn query range and partition data, and then the generated R-tree is used to execute knnJ query in parallel fashion, achieving high performance. This paper analyzes the communication and computation cost in theory. Experimental results and analysis in large real spatial data demonstrate that the algorithm can efficiently resolve the large scale knnJ spatial query in MapReduce environment, and has a good practical application.

**Key words**: cloud computing; MapReduce; *k*-nearest neighbor join; spatial query; R-tree

* 基金项目: 国家自然科学基金(61070035, 41271403); 国家高技术研究发展计划(863)(2011AA120306, 2007AA120402); 高等学校博士学科点专项科研基金(20104307110017)

收稿时间: 2012-11-12; 定稿时间: 2013-01-25

$k$-近邻连接($k$-nearest neighbor join,简称 knnJ)查询是空间数据库中一种非常重要和频繁的数据库操作,在 $k$-近邻分类、$k$-均值聚类、抽样评估、遗漏值评估等应用中,knnJ 得到了广泛的应用[1].指定空间关系 $P$ 和 $Q$,knnJ 查询为 $P$ 中每个对象返回来自 $Q$ 中 $k$ 个最近邻的对象集.knnJ 查询也是一种非常耗时的操作,需要在大量空间数据上执行昂贵的连接和最近邻计算操作.为了有效地解决 knnJ 查询问题,研究者已提出许多方法[2−4],然而,执行在单机单线程环境下,随着空间数据集的快速增长,这些方法难以扩展以应对更大规模的 knnJ 查询.

随着空间数据的急剧增长,设计云计算环境下的分布式 knnJ 查询算法越来越受到人们的关注.MapReduce[5]是 Google 提出的一种集群上处理大规模数据集的分布式并行编程模型,也是目前云计算环境中的核心计算模式.为了有效地解决大规模数据集的 knnJ 查询问题,Zhang 等人在 2012 年 ACM EDBT 会议上首先提出了 Hadoop[6]平台下的 HBNJ[7]及其改进算法 H-BRJ[7],以处理大规模多维数据的准确 knnJ 算法.H-BRJ 算法的基本思想是:阶段 1 采用块的划分方法,将输入的 $P$ 和 $Q$ 依次划分为 $n$ 块,每块对象数目约为 $|P|/n(|Q|/n)$,然后分配到 $n^2$ 个 Reducer 中进行并行处理,通过对 $Q$ 划分的子块构建 R*-树索引来加速 knnJ 查询;阶段 2 对 $R$ 的各个子块的 knnJ 查询结果进行归约,获取最终的 knnJ 查询结果.由于 H-BRJ 采用块的划分方法,其执行时间随着数据规模的增加呈平方项增长,导致查询效率不高,因此,其研究更关注近似 knnJ 查询.Wei 等人在 2012 年 ACM VLDB 会议上提出了基于 Voronoi 图数据划分的 knnJ 查询的 MapReduce 算法[8].该算法通过仔细选择 Voronoi 图的支点(pivot)来将对象分配到不同的组,每个组在一个 Reducer 内执行 knnJ 查询.尽管算法效率得到了提升,但算法受 pivot 点的选择方式和数量的影响较大.

MapReduce 框架下处理 knnJ 查询的核心在于数据的划分.本文研究 MapReduce 框架下基于 R-树索引的数据划分和任务执行算法,有效地提高了 MapReduce 框架下 knnJ 查询的效率.本文的主要贡献在于:

(1) 提出了一种支持 knnJ 查询的分布式 R-树索引快速构建算法;
(2) 提出了一种 MapReduce 框架下基于分布式 R-树索引的面向大规模空间数据集的 knnJ 查询算法,通过引入 knn 扩展框进行数据划分,限定 knn 查询范围,提高了查询效率;
(3) 通过在 32 节点的集群环境下使用真实数据集设计并完成了相关实验,实验结果和分析表明,提出的算法具有良好的效率、可扩展性和简单性,并且性能远优于 H-BRJ 算法.

本文第 1 节介绍 $k$-近邻连接查询的定义和相关背景知识.第 2 节介绍 MapReduce 框架下基于 R-树的 $k$-近邻连接算法的基本思路及查询处理算法,并从理论上进行代价分析.第 3 节给出实验结果与分析.第 4 节给出结论与展望.

## 1 问题描述和背景知识

### 1.1 查询定义

不失一般性,本文采用的距离准则为欧氏(Euclidean)距离,表示为 $d(p,q)=\left(\sum_{i=1}^{d}|p_i-q_i|^2\right)^{1/2}$.欧氏距离具有平移和旋转不变性,并满足以下 4 条性质:1) 非负性:$d(p,q)\geqslant 0$;2) 同一性:$d(p,q)=0\Leftrightarrow p=q$;3) 对称性:$d(p,q)=d(q,p)$;4) 三角不等式:$d(p,q)\leqslant d(p,r)+d(q,r)$.

**定义 1(knn 查询)**. 给定一个查询对象 $p$ 和空间关系 $Q=\{q_1,q_2,\ldots,q_N\}$,knn 查询 $knn(p,Q,k)$返回 $Q$ 中 $k$ 个 $(1\leqslant k\leqslant|Q|)$距离 $p$ 最近的对象集,并且按距离升序排列:

$$knn(p,Q,k)=\{q_1,q_2,\ldots,q_k\}.$$

其中,$q_1,q_2,\ldots,q_k\in Q;q_i\neq q_j,i\neq j$ and $\forall Q-\{q_1,q_2,\ldots,q_k\},d(p,q_j)\geqslant d(p,q_k)\geqslant d(p,q_{k-1})\geqslant\ldots\geqslant d(p,q_2)\geqslant d(p,q_1)$.

**定义 2(knnJ 查询)**. 给定一个空间关系 $P=\{p_1,p_2,\ldots,p_{NP}\}$和空间关系 $Q=\{q_1,q_2,\ldots,q_{NQ}\}$,knnJ 查询 $knnJ(P,Q,k)$返回 $P$ 中每个对象的 knn 查询结果:

$$knnJ(P,Q,k)=\{p,knn(p,Q,k)|\text{for all } p\leqslant P\}.$$

### 1.2 R-树结点的距离属性

R-树索引是一个层次化的、高度平衡的多层数据结构,是 B-树在多维数据空间上的自然扩展.R-树的每个结点对应一个最小外包框(minimum bounding rectangle,简称 MBR),该 MBR 是包围所有子结点的最小空间范围.由于本文在 R-树上执行 knnJ 查询,因此对 MBR 的一些距离函数进行定义.

设函数 $R(s,t)$表示 $E^d$ 空间中的一个 MBR,其中,$s=\{s_1,s_2,\ldots,s_d\}$为 MBR 中的最小点,$t=\{t_1,t_2,\ldots,t_d\}$为 MBR 中的最大点,即对于 $1\leqslant i\leqslant d$,有 $s_i\leqslant t_i$ 成立.在 $E^2$ 空间中,$s$ 为 MBR 的左下角点,而 $t$ 为 MBR 的右上角点.

**定义 3(MBR 间的 MinMinDist 距离)**. 给定 $E^d$ 空间中任意两个 MBR:$R_1(s,t)$和 $R_2(p,q)$,$MinMinDist(R_1(s,t),R_2(p,q))$定义为

$$MinMinDist(R_1(s,t),R_2(p,q))=\sqrt{\sum_{i=1}^{d}y_i^2},\ \text{where}\ y_i=\begin{cases}p_i-t_i, & \text{if}\ p_i>t_i\\ s_i-q_i, & \text{if}\ s_i>q_i\\ 0, & \text{otherwise}\end{cases}.$$

依据 MinMinDist 距离定义,若 $R_1(s,t)$和 $R_2(p,q)$交叠,则 $MinMinDist(R_1(s,t),R_2(p,q))$等于 0.若 $R_1(s,t)$中 $s_i=t_i$,$R_1(s,t)$退化为点 $s$,则 $MinMinDist(s,R_2(p,q))$表示 $s$ 与 $R_2(p,q)$中包围的任一对象的距离下界.

**定义 4(MBR 间的 MaxMaxDist 距离)**. 给定 $E^d$ 空间中任意两个 MBR:$R_1(s,t)$和 $R_2(p,q)$,$MaxMaxDist(R_1(s,t),R_2(p,q))$定义为

$$MaxMaxDist(R_1(s,t),R_2(p,q))=\sqrt{\sum_{i=1}^{d}y_i^2},\ \text{where}\ y_i=\max\{|p_i-t_i|,|s_i-q_i|\}.$$

MaxMaxDist 表示包含在不同 MBR 对象间的最大可能距离.图 1 描述了二维空间中两个 MBR $M_P$ 和 $M_Q$ 之间的距离函数示例.

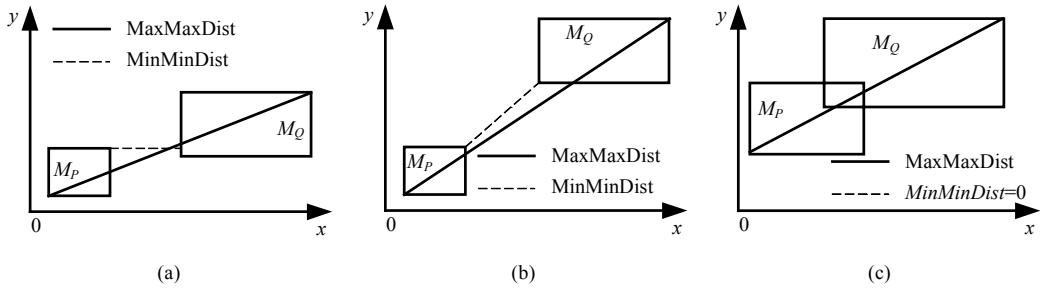



Fig.1 MinMinDist and MaxMaxDist distance between two MBR $M_P$ and $M_Q$ in $E^2$

图 1 二维空间中 MBR *MP* 和 *MQ* 之间的 MinMinDist 和 MaxMaxDist 距离

**定理 1**. 若 $E^d$ 空间中存在两个 MBR $M_P$ 和 $M_Q$,其封闭的对象子集分别为 $P=\{p_i:1\leqslant i\leqslant|P|\}$,$Q=\{q_j:1\leqslant j\leqslant|Q|\}$,则以下公式为真:

$$\forall(p_i,q_j)\in P\times Q,\ MinMinDist(M_P,M_Q)\leqslant d(p_i,q_j)\leqslant MaxMaxDist(M_P,M_Q).$$

证明:根据 MinMinDist 和 MaxMaxDist 距离的定义即可得证. □

定理 1 表明了封闭于 $M_P$ 和 $M_Q$ 内所有对象距离的下界和上界.

### 1.3 并行计算模型

Map-Reduce 编程模型通过采用一种无依赖并行和串行同步(independent parallel and serial synchronization,简称 IPSS)的计算抽象,简化了集群上大规模数据的处理.正是利用这种限制性计算抽象,MapReduce 实现了程序的自动并行处理,并在内部提供诸如输入数据划分、并行任务调度和通信、容错、负载均衡等并行分布式编程细节的实现,实现了高可扩展性和高度并行性.

**定义 5(无依赖并行计算)**. 并行计算中,给定一个作业 *J*,其可分解为一系列可异步并行执行的任务 *T*,指定

一个并行任务间的通信代价 $C$,如果对于输入任务 $T$ 的任一划分$\forall T=\{t_1,t_2,\ldots,t_m\}$,若其满足 $t_i\cap t_j=\varnothing$以及 $C(t_i,t_j)=0$ $(1<m<|T|,1<i,j<m)$条件时,作业 $J$ 可表示为 $J=\sum_{i=1}^{m}T(t_i)$, 则称满足以上条件的并行计算为无依赖并行计算.

**定义 6(串行同步计算)**. 给定两个无依赖并行计算作业 $J_1$ 和 $J_2$,若 $J_1$ 和 $J_2$ 存在一个同步和通信过程,标识为 $J_1\rightarrow_{out}d\rightarrow J_2$,即 $J_1$ 的数据输出为 $J_2$ 的输入,并且 $J_2$ 必须等到 $J_1$ 执行完毕后才开始执行,则称作业 $J_1$ 和 $J_2$ 之间是串行同步的.

MapReduce 模型是典型的 IPSS 计算抽象.如图 2 所示,Map 和 Reduce 阶段内部的若干并行任务可以在互斥的任务划分上无通信代价地独立执行,即 Map 和 Reduce 过程(称为一次 MR 过程)中每个阶段内部的异步并行任务($Map_0$~$Map_m$ 或 $Reduce_0$~$Reduce_r$)都运行在无依赖并行的理想状态下.同时,Map 和 Reduce 阶段之间是串行同步的,二者之间存在一个相对用户透明的隐式同步和通信过程.Reduce 必须等到所有 Map 执行完毕后才开始执行.但是 Map 产生中间结果的 shuffle 过程与 Map 以重叠方式执行,即任一 Map 执行完毕后,Reduce 就可以读取中间结果,这样可以缩短并行流水处理的长度,提高并行处理效率.如果一个并行应用需要多个 MR 过程,那么一次 MR 过程与下一次 MR 过程也是链式串行同步的.通信和数据交互也仅发生在一次 MR 过程中的 Map 阶段和 Reduce 阶段以及多次 MR 过程之间.

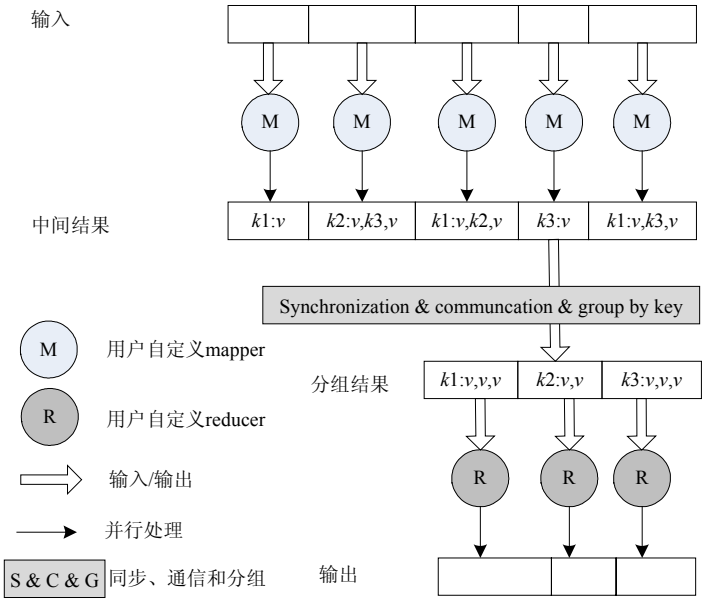



Fig.2 MapReduce parallel programming model

图 2 MapReduce 并行编程模型

目前,不少科研机构都研发了基于 MapReduce 模型的海量数据并行系统.其中,Apache 的 Hadoop 是 MapReduce 的开源实现,也是目前学术界和产业界事实上的海量数据并行处理标准.本文采用 Hadoop 作为并行计算框架.

## 2 MapReduce 框架下基于 R-树的 knnJ 查询处理

在 MapReduce 框架下利用 R-树索引来处理 knnJ 查询,其基本思想是利用 R-树索引的距离属性来进行任务划分和 knnJ 结果查询.给定参与 knnJ 的数据集依次为 $P$ 和 $Q$,其处理的基本流程可用图 3 表示.

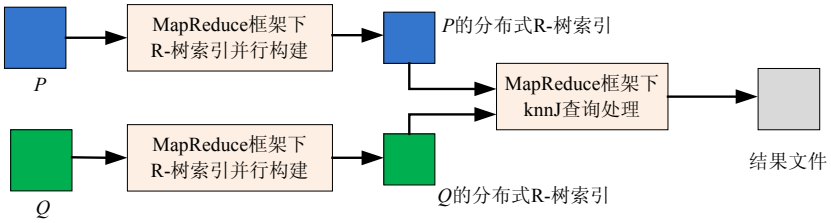



Fig.3 Basic workflow for knnJ based on R-tree in MapReduce

图 3 MapReduce 框架下基于 R-树的 knnJ 查询的基本流程

算法是在 MapReduce 的开源框架 Hadoop 平台下实现的,称为 H-rknnJ(R-tree based knnJ).H-rknnJ 查询分为两个阶段:

- 索引构建阶段:对参与连接查询的 $P$ 和 $Q$ 分别构建 R-树索引.
- knnJ 查询阶段:基于 R-树索引进行任务划分和 knnJ 查询.

不失一般性,本文涉及的空间对象由对象标识符、几何实体和其他属性数据组成,其中,几何实体属性描述对象的空间信息.对于构建的分布式 R-树索引,由子树索引地址文件和子树索引文件两部分组成.子树索引地址文件存储对应子树索引的文件地址和最小外包框 MBR,单独存放在索引文件所在目录的一个根目录文件下.子树索引文件采用 SequenceFile 文件格式进行存储,其键值对数量为 R-树索引结点的容量决定,key 表示子树索引中子结点的最小外包框 MBR(由左下角坐标$[x_l,y_l]$和右上角坐标$[x_h,y_h]$表示),value 表示子结点,采用二进制格式存储.其总的存储模型如图 4 所示.

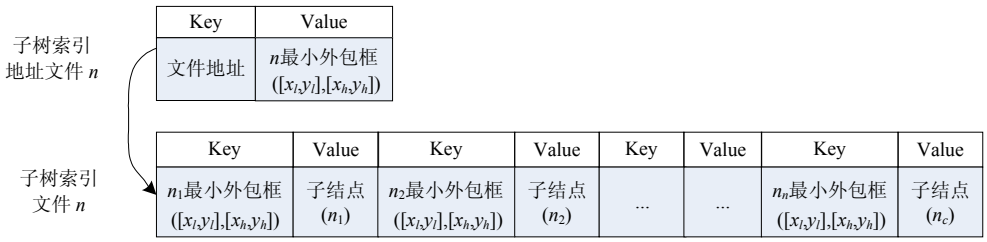



Fig.4 Key-Value pair store model for R-tree

图 4 R-树索引的键值对模型

### 2.1 MapReduce框架下R-树索引快速构建算法

在 MapReduce 框架下并行构建 R-树索引,其关键在于将空间对象均匀划分到不同分区,并尽可能地减小各个分区间空间范围的重叠面积.本文改进文献[9]中基于采样和 Hilbert 曲线的构建算法,采用两个阶段来实现 R-树索引的快速构建:

- 阶段 1:利用采样技术和 Hilbert 曲线确定空间划分函数.
- 阶段 2:分为两步:
  - ✧ 第 1 步,计算空间对象最小外包框中心点所在网格的 Hilbert 值,并将该值作为空间对象的 Hilbert 值,将每个空间对象依据空间划分函数划分到不同分区.
  - ✧ 第 2 步,将每个分区内对象按 Hilbert 值排序,按自底向上的方法构建 R-树索引.

本文将采样引入到空间数据索引的构建中,主要通过提取样本寻找切割点以确定空间划分函数,保证各个分区数据的均衡,其负载均衡和索引构建质量已在文献[9]中得到验证.利用采样生成空间对象的切割点可用图 5 进行表示,形成切割点后,空间划分函数变为

$$f(o)=\begin{cases}1, & H(o)\leqslant H'[1]\\ j, & H'[j-1]<H(o)\leqslant H'[j], j=2,...,P-1.\\ P, & H(o)>H'[P-1]\end{cases}$$

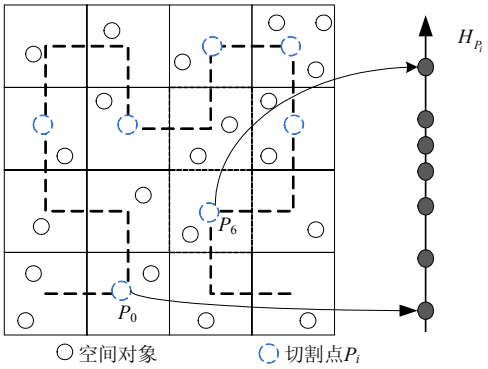



Fig.5 Generating split points

图 5 切割点生成

当空间划分函数确定时,可以证明阶段 2 中 R-树索引的构建符合 IPSS 计算模式.

**定理 2**. 阶段 2 符合 IPSS 计算模式.

证明:指定输入数据集为 $D=\{O_i,i=1,\ldots,|D|\}$,阶段 2 中第 1 步的并行计算作业 $J$ 可分解为一系列异步并行执行的任务 $T$,其中,$T$ 的划分为 $T=\{t_1,t_2,\ldots,t_{|D|}\}$,$t_i$ 表示计算第 $i$ 个空间对象 Hilbert 值的子任务,因为计算每个对象的 Hilbert 值与其他空间对象不存在关联,即 $t_i\cap t_j=\varnothing$,即 $t_i$ 和 $t_j$ 间无通信代价,有 $C(t_i,t_j)=0$ 成立.根据定义 5,第 1 步符合无依赖的并行计算模式.在第 2 步中,$D$ 划分到 $r$ 个子集,即 $D=\{D_i,i=1,\ldots,r,D_i\subseteq D\}$,满足 $\bigcup_{i=1}^{r}D_i=D$. 依据空间划分函数,空间对象的划分是单分配,即每个空间对象仅分配到其 Hilbert 值所在范围的空间划分中,因此满足 $i\neq j\Leftrightarrow D_i\cap D_j=\varnothing$.由此可以得出,每个划分是独立的.因此,$D$ 的 $R$-树索引构建可分解为一系列可异步并行执行的任务 $T$,其中,$T$ 的划分为 $T=\{t_{D_1},t_{D_2},\ldots,t_{D_p}\}$,$t_{D_i}$ 代表子集 $D_i$ 的索引构建.由前述得知,各个划分是无依赖的,因此 $t_{D_i}\cap t_{D_j}=\varnothing$ 并且有 $C(t_{D_i},t_{D_j})=0$ 成立.根据定义 5,第 2 步符合无依赖的并行计算模式,并且第 2 步必须等到第 1 步执行完毕后才开始执行,符合串行同步计算抽象.得证. □

MapReduce 框架下 R-树索引快速构建算法描述如下:

**算法 1**. MapReduce 框架下的 R-树索引构建算法.

输入:数据集 $P$,采样概率 $S_r$,子树数 $r$.

输出:子树索引集 $A$.

步骤:

1. 在计算空间划分函数的 MR 过程中,依次进行下述操作:
   1.1. $m$ 个 mappers 并行地读取输入的数据切片,对 map 函数读取的每个对象 $o$,依次进行下述操作:
      1.1.1. 若选取 $o$ 的概率 $pr(o)$满足 $pr(o)\geqslant S_r$,则计算 $o$ 中心点所在网格的 Hilbert 值,输出⟨key $c$,value *Hilbertvalue*⟩键值对,其中,$c$ 为常量;
      1.1.2. 否则,忽略此对象.
   1.2. 一个 reducer 接收所有 key 为 $c$ 的值集.reduce 函数对 Hilbert 值集进行排序,形成一个 $r-1$ 个 Hilbert 值切割点集,该切割点集将样本集分成 $r$ 个大小相等的分区.
2. 在索引构建的 MR 过程中,依次进行下述操作:
   2.1. $m$ 个 mappers 并行地读取输入的数据切片,对 map 函数读取的每个对象 $o$,利用空间划分函数计算

$f(o)$,输出⟨key $f(o)$,value ⟨*Hilbertvalue*,$o$⟩⟩键值对.

2.2. $r$ 个 reducers 接收所有划分到该 reducer 分区的对象子集,依次进行下述操作:

2.2.1. 将对象子集中每个元组⟨*Hilbertvalue*,$o$⟩插入到列表 $L$ 中.

2.2.2. 对列表 $L$ 按 Hilbert 值进行排序.

2.2.3. 按文献[9]中的串行算法构建 R-树,将结果写入如图 4 所示的键值对格式文件中,并将子树索引地址和最小外包框 MBR 写入 root 目录.

算法 1 将空间数据按空间范围划分到不同分区,然后每个分区生成一棵 R-子树索引,有利于基于距离准则的 knnJ 查询.在索引构建中,考虑到 knnJ 查询的实际需求,在构建中采用了一种基于范围限定的索引生成方法,即若构建的子树索引的空间范围大于限定范围,则算法 1 将自动将子树索引进行分裂,以避免 knnJ 查询中由于子树索引空间范围过大而造成 knnJ 查询执行过程中的内存溢出问题.

### 2.2 MapReduce框架下基于R-树索引的knnJ查询

为了实例化,本文采用如图 6 中的示例数据描述算法.图 6 中参与 knnJ 查询的索引分别为 $A$ 和 $B$.$A$ 和 $B$ 的子树索引分别为 3 个,其中,$A$ 的高度为 2,$B$ 的高度为 3.

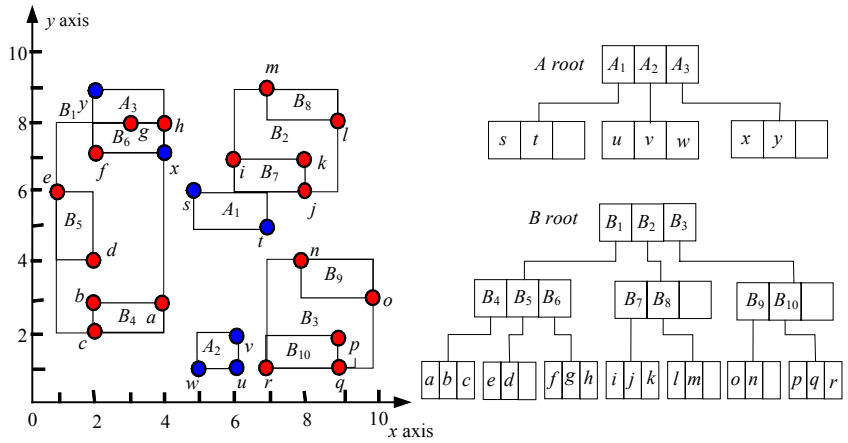



Fig.6 Example data sets and corresponding R-tree

图 6 示例数据和对应的 R-树

#### 2.2.1 基于 R-树索引的 knnJ 查询

首先描述 knnJ 的串行算法.由于 knnJ 算法基于 knn 查询,因此首先介绍 knn 查询.基于 R-树索引的 knn 查询的基本思想是:利用 MinMinDist 距离是存在最近邻的距离下界属性,不断迭代地从最近邻结点中查找最近邻对象.若空间关系 $Q$ 的 R-树索引为 $B$,给定查询点 $p$,则 R-树索引条件下的 knn 查询算法描述如下:

**算法 2**. *knn*($p$,$B$,$k$)查询算法.

输入:查询点 $p$,R-树索引 $B$.

输出:$p$ 的 $k$ 最近邻.

步骤:

1. 列表变量 $H$ 用于存储按 MinMinDist 距离排序的索引结点和对象,列表变量 $R$ 用于存储 $k$ 近邻查询结果.
2. 当初始化时,$H$ 包含 $B$ 的各个子结点及其距离。
3. 当 $H$ 中的元素不为空,并且 $R$ 中元素数量小于 $k$ 时,执行下述操作:

3.1. 从 $H$ 中提取 MinMinDist 距离最小的结点 $e$;

3.1.1. 若 $e$ 是索引结点,则将 $e$ 的所有子结点及其 MinMinDist 距离插入到 $H$ 中.

3.1.2. 否则,若 $e$ 是对象结点,则将 $e$ 及距离 $d$($p$,$e$)插入到 $R$ 中.

3.2. 从 $H$ 中删除结点 $e$.

以图 6 中示例数据和索引为例,介绍 $knn(s,B,3)$查询,以说明基于 R-树的 knn 查询过程.如图 7 所示,算法首先访问索引 $B$ 的根结点,按 MinMinDist 距离排序,然后提取 MinMinDist 距离最小的 $B_1$,并将 $B_1$ 的子结点 $B_4$,$B_5$ 和 $B_6$ 插入到 $H$ 中,并按 MinMinDist 距离排序.当提取的 $H$ 中的首元素为空间对象时,将其插入结果集 $R$ 中.重复上述操作,直至 $R$ 的势为 $k$.从中可以看出:当访问 $B_7$ 后,$H$ 中首元素为空间对象,因此其为查询点的最近邻,将其增加到 $R$ 中;当访问 $B_6$ 后,$H$ 中前两个元素均为空间对象,将其增加到 $R$ 中,此时,$R$ 的势为 3,查询满足要求,终止算法执行,$R$ 中的元素为最终的查询结果.基于 R-树索引的 knn 查询充分利用 MinMinDist 距离的下界属性,不断提取距离最小的索引结点或对象,从而获得最终的查询结果.总体来讲,基于 R-树索引的 knn 查询算法是一种增量式的查询算法,不断地提取距离最邻近的查询对象,从而获得一个比较好的查询性能.

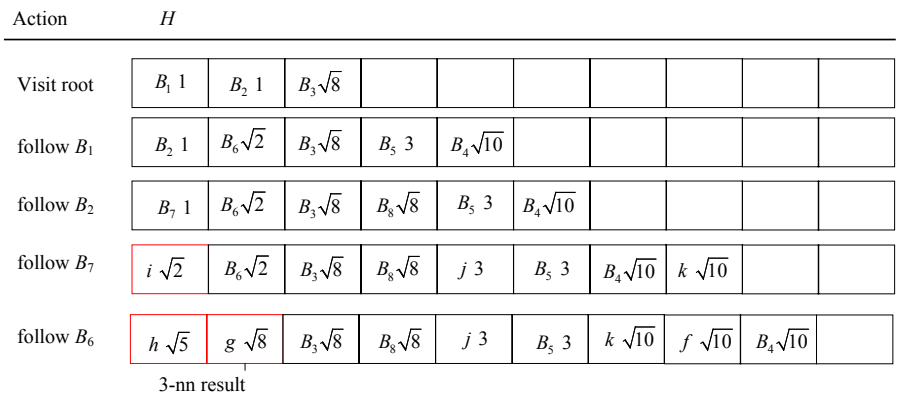



Fig.7 knn query process

图 7 knn 查询进程

若参与连接查询的空间关系 $P$ 的 R-树索引为 $A$,空间关系 $Q$ 的 R-树索引为 $B$,则 $k$-近邻连接查询 $knnJ(A,B,k)$ 可利用算法 2 来加快连接查询效率.其基本思想:对 $A$ 中的每个子结点 $e$,若 $e$ 为索引结点,则递归执行 $knnJ(e,B,k)$,直至 $e$ 为对象结点,执行 $knn(e,B,k)$.

2.2.2 基于扩展框的剪枝策略

设 $A_i$ 和 $B_j$ 分别为索引 $A$ 和 $B$ 的子树索引,在查询 $knnJ(A_i,B_j,k)$中,$A_i$ 中所有对象的 knn 查询结果的空间范围是限定的,下面利用空间范围聚集查询来计算 $A_i$ 中所有对象 knn 结果的最小外包框,称为 $A_i$ 的 knn 扩展框.

**定义 7(空间范围聚集查询**[10]**)**. 给定一个查询框 $q$ 和空间关系 $Q=\{q_1,q_2,\ldots,q_{NQ}\}$,空间范围聚集(spatial range aggregate,简称 SRA)查询 $SRA(q,Q)$返回 $Q$ 中所有位于 $q$ 内或与 $q$ 交叠的对象集的聚集值.

聚集函数包括 max,min,avg,count 等,本文主要利用 SRA 确定查询框内的对象数目,因此采用 count 函数,其算法参考文献[10].此外,本文中的 SRA 查询主要用于判断 $SRA(q,Q)$的结果是否大于 $k$,因此,在 $SRA(q,Q)$查询中,当变量 $count$ 大于或等于 $k$ 时终止查询,开始计算查询框的 knn 扩展框.

若子树索引 $A_i$ 的 MBR 为 $M_{A_i}$,空间关系 $Q$ 的 R-树索引为 $B$,则 $SRA(M_{A_i},B)$ 的查询结果分为两种情形:大于等于 $k$ 或小于 $k$,如图 8 所示,设 $M_{A_i}$ 的左下角和右上角坐标分别为 $p$ 和 $q$,其距离为 $d(p,q)$.

情形 1:当 $SRA(M_{A_i},B)\geqslant k$ 时,得出引理 1.

**引理 1**. 若 $SRA(M_{A_i},B)\geqslant k$,则 $A_i$ 中索引的所有对象的 knn 查询结果均位于扩展框 $EM_{A_i}$ 中,$EM_{A_i}$ 的左下角坐标 $p'$和右上角坐标 $q'$满足条件:

1) $p_i'=\dfrac{p_i+q_i}{2}-\dfrac{3}{2}d(p,q),i=1,\ldots,d;$

2) $q'_i=\frac{p_i+q_i}{2}+\frac{3}{2}d(p,q),i=1,...,d.$

证明:若 $SRA(M_{A_i},B)\geqslant k$, 则任意对象 $a\in A_i$,其在 $M_{A_i}$ 内存在第 $k$ 个最近邻 $b(b\in B$ 且 $b\cap M_{A_i}\neq\varnothing)$的距离上界为 $d(p,q)$.

设 $M_{A_i}$ 的质心为 $o'$,有 $o'_i=\frac{p_i+q_i}{2},i=1,...,d.$

那么,对于 $\forall b\notin EM_{A_i}$ (如图 8(a)所示),由距离的三角不等式性质有 $d(b,a)\geqslant d(b,o')-d(a,o')$成立;

又因为 $d(b,o')\geqslant\frac{3}{2}d(p,q)$ 和 $d(a,o')\leqslant\frac{1}{2}d(p,q)$ 成立,故有 $d(b,a)\geqslant d(p,q)$成立.

由此得出,任何 $EM_{A_i}$ 外的对象与 $a$ 的距离均大于等于第 $k$ 个最近邻的距离.得证. □

引理 1 保证 $M_{A_i}$ 内的所有空间对象的 knn 查询结果均在扩展框 $M_{A_i}$ 内.

情形 2:当 $SRA(M_{A_i},B)<k$ 时,向四周均匀扩张查询框范围,直至新查询框 $M'_{A_i}$ 的范围聚集查询值满足 $SRA(M'_{A_i},B)\geqslant k$, 得出引理 2.

**引理 2**. 若 $SRA(M'_{A_i},B)\geqslant k$, 则 $A_i$ 中索引的所有对象的 knn 查询结果均位于扩展框 $EM_{A_i}$ 内, $EM_{A_i}$ 的左下角和右上角坐标分别为 $p'$和 $q'$.满足条件:

1) $p'_i=\frac{p_i+q_i}{2}-(d(p,q)/2+MaxMaxDist(M_{A_i},M'_{A_i})),i=1,...,d;$

2) $q'_i=\frac{p_i+q_i}{2}+(d(p,q)/2+MaxMaxDist(M_{A_i},M'_{A_i})),i=1,...,d.$

证明:引理 2 的证明与引理 1 类似,在此不做详细论证. □

当 $SRA(M_{A_i},B)<k$ 时,采用一种均匀扩张的方法来扩张原有查询框,使得新查询框 $M'_{A_i}$ 的范围聚集查询值大于或等于 $k$(如图 8(b)所示).

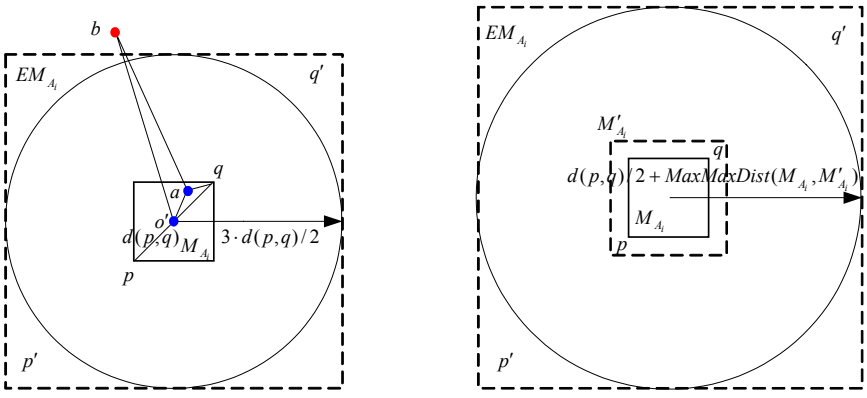
设 $k'=SRA(M_{A_i},B)$:

- 若 $k'$=0,则原有查询框向四周扩张的距离 *expDist* 计算公式为

$$expDist=(q_1-p_1)/2;$$

- 若 $k'\neq0$,则扩张距离计算公式为

$$expDist=(k-k')\times((q_1-p_1)/2\times k').$$




(a) $SRA(M_{A_i},B)$ 查询结果大于等于 $k$ (b) $SRA(M_{A_i},B)$ 查询结果小于 $k$

Fig.8 Computing knn expanding box

图 8 knn 扩展框计算

子树索引 $A_i$ 的 knn 扩展框 $EM_{A_i}$ 限定了 $A_i$ 中索引的所有对象的 knn 查询结果的距离限定范围.图 9 描述了

示例数据 A 中各个子树索引的 2-nn 查询结果的扩展框.通过计算 knn 扩展框,每个子树索引只需与另一数据集在 knn 扩展框内的对象执行 knnJ 查询即可,而不是与另一数据集的所有对象执行 knnJ 操作.

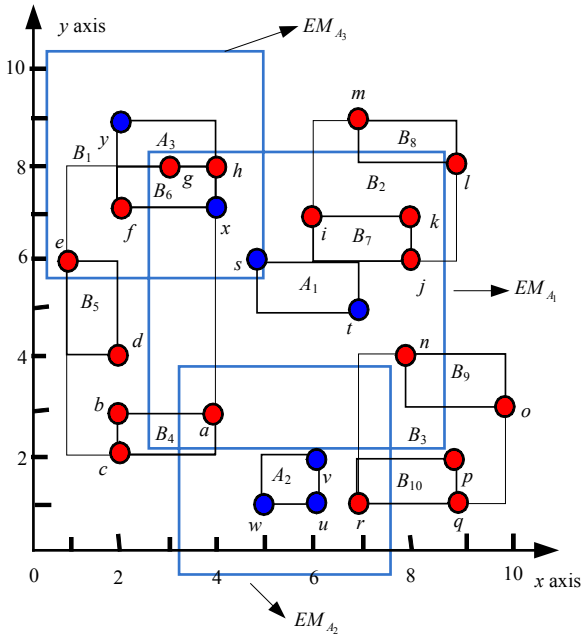



Fig.9 An example of computing knn expanding box

图 9 knn 扩展框计算示例

2.2.3 MapReduce 框架下基于 R-树索引的 knnJ 查询

在 MapReduce 框架下处理基于 R-树的 knnJ 查询处理问题,同样需要将整个计算任务分解为无依赖并行的任务集,以保证各个任务执行中不存在通信和计算依赖.

**定理 3**. 给定空间关系 $P$ 和 $Q$,其构建的索引分别为 $A$ 和 $B$, $A=\sum A_i(i=1,2,...,n), B=\sum B_j(j=1,2,...,m)$, 其中,$A_i$ 和 $B_j$ 分别表示 $A$ 和 $B$ 中的子树索引,则有 $knnJ(A,B)=\sum knnJ(A_i,B)$ 成立.

证明: 根据 R-树索引定义, $A=\sum A_i$ ($i$=1,2,…,$n$),有 $knnJ(A,B)=knnJ\left(\sum A_i,B\right)$ 成立,

根据分配律,有 $knnJ(A,B)=\sum knnJ(A_i,B)$ 成立,其中,$i$=1,2,…,$n$.证毕. □

由于 R-树的分裂性质,当 $i\neq j$ 时 $A_i\neq A_j$,因此,任务 $knnJ(A_i,B)$与 $knnJ(A_j,B)$间没有数据和通信依赖.根据上述分析,$knnJ(A,B)$可分解为无依赖并行执行的任务集,因此可用 MapReduce 框架实现.

更进一步地,依据引理 1 和引理 2 给出定理 4.

**定理 4**. 给定参与 knnJ 的索引 $A$ 和 $B$,$A_i$ 和 $B_j$ 分别 $A$ 和 $B$ 中的子树索引,$A_i$ 的扩展框为 $EM_{A_i}$, 则

$$knnJ(A,B)=\sum knnJ(A_i,B_{EM_{A_i}}),$$

其中, $B_{EM_{A_i}}$ 表示 $B$ 中所有位于扩展框 $EM_{A_i}$ 内的对象集的 R-树索引.

下面依据上述的分析和无依赖并行任务划分方法,详细描述 MapReduce 框架下的 knnJ 查询算法 H-rknnJ.

**算法 3**. H-rknnJ 查询算法.

输入:分布式 R-树索引 $A$ 和 $B$.

输出:$A$ 中索引的每个对象的 $k$ 最近邻.

步骤:

1. 在 Master 节点提交作业时,依次执行下述操作:

1.1. 将 $A$ 中包含每个子树索引地址及其 MBR 的文件写入 $m$ 个输入文件,指定为 Map 阶段的输入.

1.2. 将 $B$ 中所有包含子树索引的文件地址 *ptr* 和最小外包框 MBR 合并成一个文件,利用分布式缓存技术将其分发到各个计算节点.

2. 在 Map 阶段中,依次执行下述操作:

2.1. 在 Map 阶段的 setup 操作中,执行下述操作:

2.1.1. 初始化列表变量 *bList*.

2.1.2. 读取缓存文件,将缓存文件中 $B$ 的每个子树索引地址 *ptr* 和 MBR 插入到 *bList* 中.

2.2. 在 Map 阶段的 map 操作中,输入 value 为 $A$ 中子树索引 $A_i$ 的地址 *ptr* 和及其最小外包框 MBR,执行下述操作:

2.2.1. 初始化整数变量 *count* 为 0.

2.2.2. 依次取 *bList* 中的元素 $t$,若 $t$ 的 MBR 与 $A_i$ 的 MBR 交叠,依据 *t.ptr* 读取 $B$ 中的子树索引 $n$,执行范围聚集查询 $SRA(A_i.MBR,n)$,并将结果累加到 *count* 中:若 *count* 大于或等于 $k$,则终止操作,执行下一步;否则,继续取 *bList* 中的元素,执行范围聚集查询,直至满足 *count* 大于或等于 $k$ 或 *bList* 所有元素查询结束.

2.2.3. 若 *count* 大于或等于 $k$,基于引理 1 计算 knn 扩展框 $EM_{A_i}$.

2.2.4. 否则,基于引理 2 的查询框扩张规则扩张查询框范围,直至新查询框的范围聚集值 *count* 大于或等于 $k$,基于引理 2 计算 knn 扩展框 $EM_{A_i}$.

2.2.5. 以 $EM_{A_i}$ 为查询框,依次取 *bList* 中的每个元素,若与 $EM_v$ 交叠,则读取子树索引,执行空间范围查询,并将结果插入到集合 $S$ 中.

2.2.6. 为 $S$ 构建 R-树索引 *Ns*.

2.2.7. 依据 $A_i$ 的 *ptr* 读取 $A$ 中的子树索引 $A_i$.

2.2.8. 执行 $knnJ(A_i,Ns,k)$算法.

与 H-BRJ 中基于块的数据划分不同,H-rknnJ 算法是一种主动“抓取”式的数据划分方式,在访问 $B$ 中的子树索引时,采用 HDFS 中的文件复制方法,将需要访问的子树索引快速复制到本地的工作目录中,以避免索引并发访问造成的性能瓶颈.在算法执行过程中,H-rknnJ 算法将已构建好的 $A$ 中每个子树索引的文件地址和最小外包框作为 map 函数的输入,map 函数通过子树索引外包框从缓存的 $B$ 的根结点中查找与其交叠的子树索引,通过执行范围聚集查询计算该子树索引中所有对象的 knn 查询结果的扩展框,然后以扩展框作为范围查询的输入条件,执行范围查询,抓取 $B$ 在扩展框内的所有对象集 $S$,并为 $S$ 构建 R-树索引.最后,map 函数读取 $A_i$ 中对应的子树索引并执行 knnJ 查询.

### 2.3 代价分析

在 H-rknnJ 算法中,

- 阶段 1 利用采样和 Hilbert 曲线来计算空间划分函数,采样概率为 $S_r$,因此,通信代价为 $O(|P|S_r+|Q|S_r)$,计算代价为计算样本点 Hilbert 值的代价 $O(|P|S_r+|Q|S_r)$,排序代价 $O(|P|S_r\log(|P|S_r)+|Q|S_r\log(|Q|S_r))$,选取 $r$ 个切割点的代价为 $O(2r)$.
- 阶段 2 利用空间划分函数将 $P$ 和 $Q$ 分别划分为 $r$ 块,Reducer 读取 $r$ 个子块,因此其通信代价为 $O((|P|/r+|Q|/r)r)$,化简处理得 $O(|P|+|Q|)$.计算每个对象 Hilbert 值的代价 $O(|P|+|Q|)$.在索引构建中,对 $P$ 中的子块 $P_{bi}$ 构建索引的代价为 $O(|P_{bi}|\log|P_{bi}|)$,又由于$|P_{bi}|=|P|/r$,且构建的子块数量为 $r$,因此,$P$ 的构建代价为 $O(|P|\log(|P|/r))$,则 $P$ 和 $Q$ 总的构建代价为 $O(|P|\log(|P|/r)+|Q|\log(|Q|/r))$.
- 阶段 3 利用 R-树索引来执行 knnJ 查询,将 $P$ 中的每个子树索引 $A_i$ 地址和最小外包框作为 $r$ 个 mappers 的输入,mapper 从 HDFS 中读取子树索引 $A_i$ 及 $Q$ 在 $A_i$ 的扩展框内的空间数据,若 $P$ 和 $Q$ 同分布,则依据扩展框的计算理论,扩展框内 $Q$ 的空间对象数目为 $A_i$ 的 $c$ 倍(在 $E^2$ 空间中,$c$>9),其通信代价为

$O(|A_i|r+c|Q|)$,由于$|A_i|=|A|/r$,因此,通信代价可代简为 $O(|A|+c|Q|)$.考虑内存约束问题,算法只读取 $Q$ 在扩展框内空间对象集并构建索引,其索引构建代价可表示为 $O(c|Q|\log(c|Q|/r))$,$A_i$ 索引的对象数为$|P_{bi}|$,计算 $A_i$ 中每个对象在扩展框内 knn 结果集的计算代价为 $|P_{bi}|\sqrt{c|Q_{bj}|}$, 总的 knnJ 计算代价为 $O\left(\left(|P|/r\sqrt{c|Q|/r}\right)r\right)$, 化简为 $O\left(|P|\sqrt{c|Q|/r}\right)$.

依据上述分析,H-rknnJ 总的通信代价为 $O((1+S_r)|P|+(1+c+S_r)|Q|+|A|)$,忽略抽样概率 $S_r$,总的通信代价可表示为 $O(|P|+(1+c)|Q|+|A|)$.考虑计算代价时抽样阶段可忽略不计,索引构建和 knnJ 查询阶段的计算代价可表示为

$$O\left(\left(1+\log(|P|/r)+\sqrt{c|Q|/r}\right)|P|+(1+(1+c)\log(c|Q|/r))|Q|\right).$$

与 H-BRJ 算法相比,其通信代价将由 $O(n(|P|+|Q|)+nk|P|)$减少至 $O(|P|+(1+c)|Q|+|A|)$,其通信代价明显得到大幅度少.而 H-BRJ 算法的计算代价是 $O\left(n|Q|\log(|Q|/n)+n|P|\sqrt{|Q|/n}+|P|nk\log k\right)$, H-rknnJ 算法的计算代价中缺少倍率参数 $n$,因此,其计算代价同样得到优化.

## 3 实验结果与分析

### 3.1 实验内容与设置

Hadoop 集群包括 1 台主控节点和 32 个计算节点.每个节点配置为 2 quad-core 2.4GHZ CPU,16GB 内存和 167GB 的本地存储磁盘.操作系统为 Red Hat Enterprise Linux 5.5,节点间通过 10-Gigabit Infiniband 网络互联,MapReduce 框架基于 Hadoop 平台 0.20.1,每节点上可并行运行的 Mapper 数和 Reducer 数均设为 4,其他采用默认设置.

为与文献[7]中实验内容保持一致,本文数据同样采用来自 OpenStreetMap 项目的真实数据集.每一个数据集是美国一个州的道路网数据,美国 50 州的完整数据集记录数大于 160M.每条记录包含一个记录标识 ID、二维坐标和其他描述信息,采用 XML 文件存储,其大小为 82GB.在实验的预处理阶段,需要将 XML 文件格式中对象 ID、二维坐标提取出来,采用一个简单的 MapReduce 程序即可实现,提取出的数据集大小为 13GB.为测试算法性能,并与其他算法进行性能对比,采用基于 MapReduce 的随机采样算法从数据集中分别提出两个样本集 $P$ 和 $Q$,其中,$P$ 和 $Q$ 的数据规模变化为 10M,20M,40M 和 80M.采用($M$×$N$)标识数据配置,其中,$M$ 和 $N$ 分别表示 $P$ 和 $Q$ 中记录数(百万),如(40×80)表示有 40M 的 $P$ 记录数和 80M 的 $Q$ 记录数.

实验内容利用上述的实验环境和实验数据来测试提出的 H-rknnJ 算法的性能,并与 H-BRJ 算法进行性能对比.Hilbert 的阶设为 15,即将整个空间连接区域划分为 $2^{15}\times2^{15}$ 个网格,基于此网格划分来构建 R-树索引,实验从数据规模 $M$×$N$、节点数 $N$ 和 $k$ 值这 3 个方面测试算法的性能影响.参数设置见表 1.

**Table 1** Experimental parameters

**表 1** 实验参数设置

| 参数 | 数据规模/$10^6\times10^6$ | 节点数 | $k$ |
|---|---|---|---|
| 默认值 | 40×40 | 32 | 5 |
| 变化范围 | 10×10~80×80 | 4~32 | 1~20 |

### 3.2 索引构建性能

R-树索引构建是执行 H-rknnJ 算法的初始步骤.实验在 32 节点的集群上依次对 $P$ 和 $Q$ 分别构建索引.图 10 和图 11 分别描述了 $P$ 和 $Q$ 中不同规模数据集的构建时间,其中,

- 阶段 1 为抽样阶段,形成空间划分函数,其时间在 27s~29s 范围内变化.
- 阶段 2 为 R-树索引构建时间.当数据规模为 10M 时,R-树索引的构建时间为 48s;当数据规模为 80M 时,构建时间最多为 73s.即使关系 $P$ 和 $Q$ 的构建为串行关系,当数据规模为 80M 时,其总的构建时间最多为 204s.因此,R-树索引的构建过程非常高效.

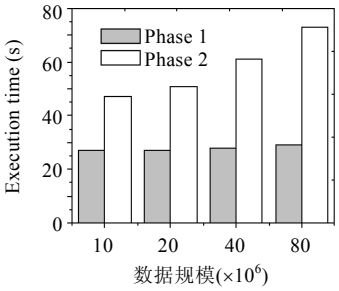



Fig.10 R-Tree index constructing performance for $P$

图 10 $P$ 的 R-树索引构建性能

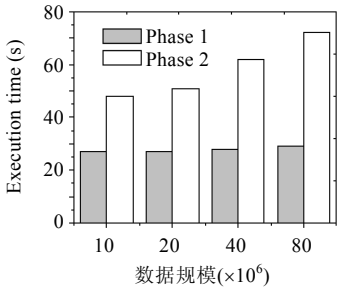



Fig.11 R-tree index constructing performance for $Q$

图 11 $Q$ 的 R-树索引构建性能

### 3.3 数据规模的效果

下面测试数据规模对 H-rknnJ 算法的性能影响.实验从两个方面进行:首先,固定计算节点数 $N$=32,测试不同 $k$ 值条件下不同数据规模的时间性能;其次,固定 $k$=5 值,测试不同 $N$ 值条件下不同数据规模算法的时间性能.图 12 表示固定 $N$ 值下的实验结果.从图中可以看出,在不同 $k$ 值条件下,随着 $M$×$N$ 的增长,其查询时间随数据规模的增长呈近线性增长.从中可以看出:当 $k$=5 时,40×40 的执行时间是 350s,而 80×80 的执行时间是 554,其线性比为 1.58,尽管其计算任务增长了 4 倍,但其执行时间仅增长了 1.58 倍,表现出非常好的可扩展性能;当 $k$ 为其他值时,算法表现出类似的结果.图 13 表示固定 $k$=5 值下的实验结果.从中可以看出:在不同 $N$ 值条件下,随着 $M$×$N$ 的增长,其查询时间同样呈近线性增长.$N$ 越小,执行时间增长得越陡峭;$N$ 越大,执行时间增长得越平缓.主要原因在于,当 $N$ 越大,数据规模相对较小时,计算资源并没有得到充分利用,因此,尽管数据规模在增长,但其执行时间增长并不快.如图中当 $N$=32,数据规模从 10×10 变化到 20×20 时,其执行时间仅增长了 1.29;相反地,当 $N$=4,数据规模从 10×10 依次增长到 80×80 时,其执行时间依次增长了 1.8 倍、3.54 倍和 6.73 倍,表明集群中的计算资源得到了充分的利用.

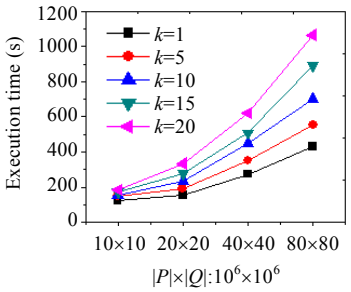



Fig.12 $N$ is fixed, the time performance for variable $k$

图 12 固定 $N$ 值,不同 $k$ 值的时间性能

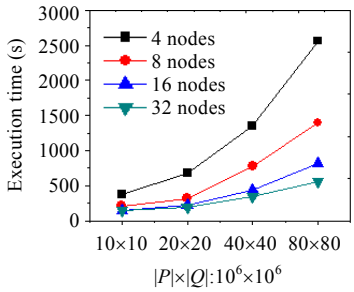



Fig.13 $k$ is fixed, the time performance for variable $N$

图 13 固定 $k$ 值,不同 $N$ 值下的时间性能

### 3.4 $k$的效果

下面测试 $k$ 值对 H-rknnJ 算法的性能影响.实验从两个方面进行:首先,计算节点采用默认设置,测试不同数据规模条件下不同 $k$ 值对算法性能的影响;其次,数据规模采用默认设置,测试不同 $N$ 条件下随 $k$ 值的性能变化情况.图 14 表示当 $N$ 固定时,$k$ 值在不同数据规模下对算法的性能影响.从中可以看出,在不同数据规模条件下,随着 $k$ 值的增长,算法执行时间呈近线性增长.图 15 表示数据规模固定的情况下,$k$ 值在不同 $N$ 值条件下对 H-rknnJ 算法的性能影响.随着 $k$ 值的增加,算法的执行时间增加.其原因在于,$k$ 值对 H-rknnJ 算法性能的影响是

两个方面的:1) $k$ 增加,则对于 $P$ 中子树索引参与 knnJ 查询的 knn 扩展框的空间范围增大,则需要访问 $Q$ 中更多的子树索引,增加了索引访问时间;2) $k$ 增加,则 $P$ 中子树的数据结点获取 knn 查询结果的时间增加,增加了总体的查询时间.

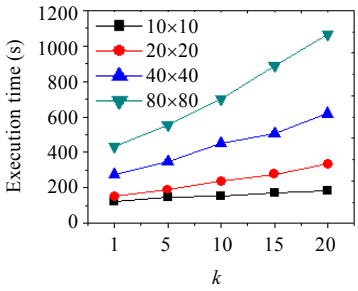



Fig.14 $N$ is fixed, the effect of $k$

图 14 固定 $N$ 值,$k$ 的效果

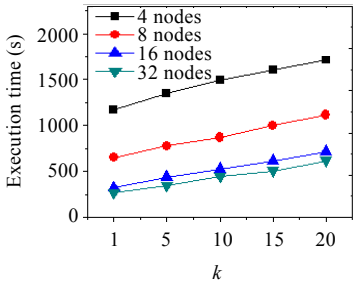



Fig.15 $M\times N$ is fixed, the effect of $k$

图 15 固定 $M\times N$ 值,$k$ 的效果

### 3.5 $N$的效果

下面测试 $N$ 值对 H-rknnJ 算法性能的影响.实验从两个方面进行测试:实验 1 固定 $k$=5,测试不同数据规模条件下算法随 $N$ 值对算法的性能影响;实验 2 固定数据规模为 40×40,测试不同 $k$ 值条件下 $N$ 值对算法性能的影响.图 16 表示实验 1 的实验结果,图 17 表示的是实验 2 的实验结果.从图中可以看出,随着 $N$ 的增长,算法的执行时间均呈近线性递减,表现出良好的可扩展性.计算节点是衡量可扩展性的一个重要维度,若计算任务不变,当计算节点增加 1 倍,则执行时间应当减少 1 半.然而在实际算法运行过程中,受容错、数据通信等各方面因素的影响,其执行时间以一种近线性的速率递减.从两图中还可观察出,当计算节点从 16 增加到 32 时,执行时间递减得更加平缓.主要原因在于 16 个计算节点足以满足不同计算任务条件下 H-rknnJ 的查询性能,若增加到 32 个计算节点,则会导致部分计算资源空闲的情况发生,因此,其执行时间递减得更加平缓.

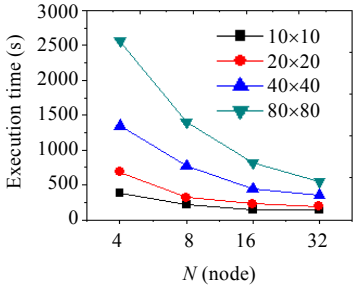



Fig.16 $k$ is fixed at 5, the effect of $N$

图 16 固定 $k$=5 值,$N$ 的效果

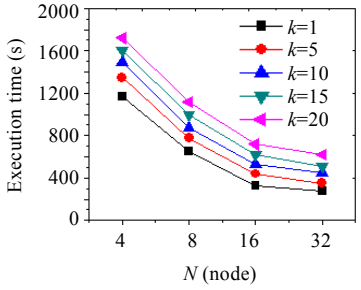



Fig.17 $M\times N$ is fixed, the effect of $N$

图 17 固定 $M\times N$ 值,$N$ 的效果

### 3.6 与H-BRJ算法的性能对比

H-BRJ 算法是数据库会议 ACM EDBT2012 会议上提出的 MapReduce 框架下面向大规模数据的 knnJ 查询算法.虽然 H-BRJ 算法效率与串行算法相比得到较大的提升,但是 H-BRJ 采用基于块的划分方法,其执行时间随着数据规模的增加呈平方项增长,导致查询效率并不高.因此,在进行性能对比时,固定 $k$ 值为 5,$N$ 为 32.其中,H-rknnJ 执行时间将 $P$ 和 $Q$ 的索引构建时间包含在内.图 18 表示 H-rknnJ 与 H-BRJ 在数据规模为 10×10 条件下,不同 $k$ 值条件下算法性能的对比.从中可以看出,H-rknnJ 算法性能明显优于 H-BRJ,当 $k$ 从 1 变化至 20 时,H-BRJ 算法的执行时间依次是 H-rknnJ 算法的 4.28 倍~6.5 倍.图 19 表示 H-rknnJ 与 H-BRJ 当 $k$=5 时,在不同数据规模下算法性能的对比.由于 H-BRJ 算法效率低下,因此数据规模仅从 10×10 变化至 40×40.显然,H-rknnJ 采

用基于扩展框的任务划分,只需要与扩展框内的对象进行 knnJ 查询,避免了与 $Q$ 中所有子块进行 knnJ 查询;同时,不仅不需要 H-BRJ 算法中阶段 2 的归约处理,并且所有查询过程只需要在一次 Map 过程中完成.因此,算法性能表现出良好的线性可扩展性.相反地,H-BRJ 采用块的划分,其执行时间呈平方增长,并且其执行时间远远高于 H-rknnJ.当数据规模从 10×10 变化至 40×40 时,执行时间依次增长了 4.35 倍、8 倍和 14.3 倍,接近平方增长,与前面的理论分析符合.此外,由于文献[8]中的算法没有发布且难以实现,故本文并没有与其进行性能对比.

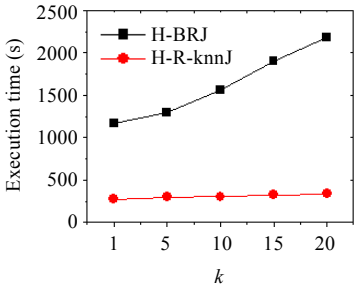



Fig.18 Performance comparison with H-BRJ when $M\times N$ is fixed at 10×10

图 18 固定 $M\times N$=10×10 时的性能对比

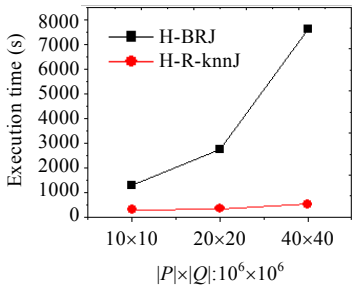



Fig.19 Performance comparison with H-BRJ when $k$ is fixed at 5

图 19 固定 $k$=5 时的性能对比

## 4 结 论

本文针对 MapReduce 框架下处理大规模空间数据的 knnJ 查询问题,首先提出了一种 MapReduce 框架下 R-树索引的快速构建算法,利用 R-树索引的构建结果,提出了一种 MapReduce 框架下的空间 knnJ 查询算法 H-rknnJ,通过引入 knn 扩展框限定子树索引的 knnJ 查询范围,有效地减少了查询中所需要通信和计算的代价.H-rknnJ 算法能够高效地处理大规模空间数据的 knnJ 查询问题,具有良好的可扩展性和查询效率.实验结果和分析表明, H-rknnJ可以很好地支持大规模空间数据的knnJ查询处理,具有很好的实用价值.H-rknnJ算法不仅面向 knnJ 查询,而且只需要简单的扩充,即可应用于其他空间相似性连接查询上,如距离连接、反 knnJ 查询.特别地,在性能对比上,H-rknnJ 算法的性能远优于 H-BRJ 算法.此外,本文的研究工作主要限定于地理空间上基于欧氏距离的knnJ查询,没有测试高维空间和不同度量准则对算法性能的影响,因此,下一步需要研究MapReduce框架下面向高维空间的 knnJ 查询算法,进一步优化算法性能,提高 H-rknnJ 算法的适用范围.

**References**:

[1] Bohm C, Krebs F. The *k*-nearest neighbor join: Turbo charging the KDD process. Knowledge Information System, 2004,6(6): 728−749. [doi: 10.1007/s10115-003-0122-9]

[2] Xia CY, Lu HJ, Coi BC, Hu J. Gorder: An efficient method for KDD joins processing. In: Proc. of the 30th Int'l Conf. on Very Large Data Bases (VLDB). 2004. 756−767.

[3] Yao B, Li FF, Kumar P. *K* nearest neighbor queries and KNN-joins in large relational databases (almost) for free. In: Proc. of the 26th Int'l Conf. on Data Engineering (ICDE). 2010. 4−15. [doi: 10.1109/ICDE.2010.5447837]

[4] Yu C, Cui B, Wang SG, Su JW. Efficient index-based KNN join processing for high-dimensional data. Information and Software Technology, 2007,49(4):332−344. [doi: 10.1016/j.infsof.2006.05.006]

[5] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. Communications of the ACM, 2008,51(1):107−113. [doi: 10.1145/1327452.1327492]

[6] White T. Hadoop: The Definitive Guide. Sebastopol: Yahoo! Press, 2009.

[7] Zhang C, Li FF, Jestes J. Efficient parallel kNN joins for large data in MapReduce. In: Proc. of the 15th Int'l Conf. on Extending Database Technology (EDBT). 2012. 38−49. [doi: 10.1145/2247596.2247602]

[8] Lu W, Shen YY, Chen S, Coi BC. Efficient processing of *k* nearest neighbor joins using MapReduce. In: Proc. of the 38th Int'l Conf. on Very Large Data Bases (VLDB). 2012. 1016−1027.

[9] Liu Y, Jing N, Chen L, Chen HZ. Parallel bulk-loading of spatial data with MapReduce: An R-tree case. Wuhan University Journal of Natural Sciences, 2011,16(6):513−519. [doi: 10.1007/s11859-011-0790-3]

[10] Tao YF, Papadias D. Range aggregate processing in spatial databases. IEEE Trans. on Knowledge and Data Engineering, 2004, 16(12):1555−1570. [doi: 10.1109/TKDE.2004.93]

**刘义**(1979－),男,湖南华容人,博士生,CCF 学生会员,主要研究领域为空间数据库,地理信息系统.
E-mail: liu.yi.nudt@gmail.com

**陈荦**(1973－),男,博士,教授,CCF 会员,主要研究领域为地理空间信息处理.
E-mail: luochen@nudt.edu.cn

**景宁**(1963－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据库系统,地理信息系统,空天资源规划决策技术,空间数据可视化技术.
E-mail: ningjing@nudt.edu.cn

**熊伟**(1976－),男,博士,副教授,CCF 会员,主要研究领域为空间数据库,地理信息系统.
E-mail: xiongwei@nudt.edu.cn