第48卷第1期 中国科学技术大学学报 Vol.48,No.1
2018年1月 JOURNAL OF UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA Jan. 2018

文章编号:0253-2778(2018)01-0047-10

一种基于轨迹数据密度分区的分布式并行聚类方法

王佳玉[1],张振宇[1,2],褚 征[1],吴晓红[2]

(1.新疆大学软件学院,乌鲁木齐 830008;2.新疆大学信息科学与工程学院,乌鲁木齐 830046)

摘要:全球定位技术与基于位置服务的发展促进了轨迹大数据的发展.轨迹聚类作为最重要的轨迹分析任务之一,得到了广泛的研究.目前,大多数聚类方法是在单处理机模式下运行,对于大规模的轨迹数据其处理时间较长,难以满足时效性强的轨迹分析任务,为此提出一种基于轨迹数据密度分区的分布式并行聚类方法.首先将整个轨迹数据集抽象在一个矩形区域内,通过该矩形最长维度的变换将数据合理地划分为若干任务量相当的分区,构建可供分布式并行聚类的局部数据集,然后各工作服务器对局部分区分别执行DBSCAN聚类算法,管理服务器对局部聚类结果进行合并与整合.实验结果验证了本方法的有效性,在一定程度上提高了聚类分析的运算效率.

关键词:轨迹大数据;分布式聚类;DBSCAN算法;聚类算法

中图分类号:TP391 **文献标识码**:A doi:10.3969/j.issn.0253-2778.2018.01.007

引用格式:王佳玉,张振宇,褚征,等.一种基于轨迹数据密度分区的分布式并行聚类方法[J].中国科学技术大学学报,2018,48(1):47-56.

WANG Jiayu, ZHANG Zhenyu, CHU Zheng, et al. A trajectory data density partition based distributed parallel clustering method[J]. Journal of University of Science and Technology of China, 2018,48(1):47-56.

A trajectory data density partition based distributed parallel clustering method

WANG Jiayu[1], ZHANG Zhenyu[1,2], CHU Zheng[1], WU Xiaohong[2]

(1. *School of Software, Xinjiang University, Urumqi* 830008;
2. *College of Information Science and Engineering, Xinjiang University, Urumqi* 830046)

Abstract: The development of global positioning technology and location-based service have contributed to the development of trajectory big data. Trajectory clustering is one of the most important trajectory analysis tasks and has been extensively studied. Currently, most of the clustering methods operate in a single-processor mode, and large-scale trajectory data processing is a lengthy process, making it difficult to meet the strong timeliness of the trajectory analysis task. To solve the problem, a distributed parallel clustering method based on trajectory density partition is proposed. Firstly, the whole dataset is abstracted in a rectangular region, and the dataset is divided into several partitions with tasks that have almost the same amount by the transformation of the longest dimension of the rectangle, thus constructing the local datasets for distributed parallel clustering. Then the worker servers implement the DBSCAN clustering algorithm for the local partitions respectively, and the manager server merges and integrates the local

收稿日期:2017-05-20;**修回日期**:2017-06-23

基金项目:国家自然科学基金项目(61262089)资助.

作者简介:王佳玉,女,1991年生,硕士生,研究方向:移动计算,E-mail:jennywang91@126.com

通讯作者:张振宇,教授,E-mail:zhangzhenyu@xju.edu.cn

clustering results. The experimental results show that the algorithm is effective and improves the computational rate of clustering analysis to a certain degree.

Key words: trajectory big data; distributed clustering; DBSCAN algorithm; clustering algorithm

0 引言

随着互联网技术和相关应用需求的蓬勃发展，智能手机、可穿戴设备、个人手持设备(personal digital assistant，PDA)等逐渐成为人们日常生活和工作中的必备品.与此同时，智能设备普遍配备了全球定位系统(global positioning system，GPS)传感器，通常会在一些应用平台上产生大量的由坐标构成的空间位置数据，这些数据组成了蕴含丰富信息的移动轨迹.这些潜在信息通常包括用户的爱好和行为习惯等.通过综合分析移动用户的轨迹信息，可以发掘出某一地域的热点区域，了解人类活动与地域之间的关系，能更好地为用户提供服务.轨迹聚类分析是对移动用户社会角色发现的有效方法，也是轨迹数据挖掘的重要研究内容[1-3].

目前，针对移动轨迹数据挖掘的研究已经取得了相当大的进展.文献[4]提出了针对不同轨迹间点对距离的分层聚类算法.文献[5]提出一种改进的基于密度的航迹聚类算法，采用加权 Manhattan 距离与惩罚系数相结合的距离度量，根据目标运动的特征自定义点的邻域，利用时间裁剪提高算法运行效率.文献[6]利用索引树存储轨迹数据和其相似矩阵来完成有效的轨迹信息聚类过程.文献[7]提出了一种可以使用任意 n 维可用数据来检测集群的聚类方法.文献[8]提出了一个基于 partition-and-group 框架的轨迹聚类算法 TRACLUS，通过先分割再分组的方法有效地从繁杂的轨迹数据库中找到公共子轨迹.文献[9]将各种聚类的有效性指标作为优化的目标函数，利用粒子群优化的方式实现了轨迹聚类的自动化技术，能够有效地在聚类过程中找到最佳聚类数量及聚类中心.

移动轨迹数据随时间增长及用户量的增多，其增长速度呈现海量态势，且轨迹覆盖区域辽阔，这就大大增加了数据分析的时间和空间复杂度；同时，一些具有时效性的基于位置的服务其性能也必将受影响.以上这些聚类方法都是在单处理机模式下运行且没有考虑应用的时效性要求，计算时间通常较长，难以适应规模庞大、时效性强等轨迹数据特点所带来的挑战.

本文提出了一种基于轨迹数据密度分区的分布式并行聚类方法.该方法分为两个阶段，首先根据轨迹数据中位置坐标的空间分布特性，将整个数据空间抽象成一个矩形，然后将其划分为若干个局部矩形分区，再对这些局部区域实行分布式并行计算，以此来提高轨迹数据聚类分析的处理速率.

本文的主要贡献如下：

(Ⅰ)提出分布式并行聚类算法，并构建了该算法的分布式架构原型，通过对聚类算法进行分布式处理，有效缓解了串行聚类 CPU 占用过高、内存高耗的情况；

(Ⅱ)提出了基于数据密度的分区算法，能够将整个数据集合进行合理的分区，保证了在局部聚类时各任务量相当；

(Ⅲ)本文提出的聚类方法解决了海量轨迹数据聚类分析延时高的问题.

1 相关工作

轨迹数据的聚类旨在通过基于轨迹的运动特性将其分到有限的类别(簇)中来描述轨迹数据集，同一个簇中的轨迹数据具有相同的特性.目前广泛应用的聚类方法有多种，主要分为基于划分的方法、基于层次的方法、基于密度的方法以及基于网格的方法[10]等.

1.1 基于划分的方法

基于划分的方法也称为基于目标函数的聚类算法，是一类常见的聚类方法[11].划分方法是指将给定的数据集通过一定的规则或方法划分成用户事先设定的 k 个组或簇，每个组中都应该至少包括一个数据对象，并且，每个组之间的数据具有显著的不同，同一个组中的数据要尽可能相似或相关.划分方法的聚类过程实际上是将聚类问题转化为优化问题，通过优化一个评价函数将数据逐步划分到组中，划分法形成的每个组中的数据具有很强的相似性，便于进行整体分析.常见的基于划分的聚类方法有 k-means 算法[12]，该算法是一种快速的聚类算法，对大数据有较高的效率，但是由于需要预先设定 k 值及初始化中心节点，且只能发现球形的簇，局限了该算法的应用.

1.2 基于层次的方法

基于层次的方法通过创建一个层次来分解给定的数据集，该层次是由若干组组成的树形结构.由于层次构建的方式的不同，该方法可以分为自底向上(凝聚)的方法和自顶向下(分裂)的方法.在采用层次聚类的方法时，通常会面临分解与合并的不足问题，所以层次合并经常要与其他聚类方法结合使用.CURE 算法[13]利用固定数目代表对象来表示相应聚类，然后对各聚类按照指定量向聚类中心进行收缩，利用随机抽样与分割相结合的办法来提高算法的空间和时间效率.BIRCH 算法[14]首先利用树的结构对对象集进行划分，然后再利用其他聚类方法对这些聚类进行优化.BIRCH 算法只适用于类的分布呈凸形及球形的情况，并且由于 BIRCH 算法需提供正确的聚类个数和簇直径限制，对不可视的高维数据不可行。

1.3 基于密度的方法

大部分的聚类算法主要是基于对象间的距离进行聚类，这种聚类方法在面临任意形状的簇时不一定能有效地聚类，因此提出基于密度的聚类算法.该方法的核心思想是：相邻局域的密度(对象数据的个数)达到某个预先设定的阈值，就把该对象加入到此簇中，并持续聚类直到每个点都加入到相应的簇.该方法过滤了“噪声”，去除了异常点，从而降低了时间复杂度，提高了数据聚类分析的效率.DBSCAN 算法[15]是一种经典的基于密度的聚类算法.很多学者在此算法的基础上做了相关研究.文献[16]提出了一种快速的 DBSCAN 聚类算法，利用其提出的基于图形的索引结构方法及 Groups 方法加速了邻居搜索操作.文献[17]使用模糊集理论改进了 DBSCAN 算法，改进后的算法能够有效处理信息含糊的数据资料的收集与分析，但不适宜于轨迹数据的分析.

1.4 基于网格的方法

该方法将数据对象划分为有限数目的单元，从而形成一个网格，在网格上进行聚类操作.基于网格方法的处理时间独立于数据对象的数目，能够有效地提高处理速度.但基于网格的方法处理时间与每维空间所划分的单元数相关，一定程度上降低了聚类的质量和准确性.文献[18]提出了不确定度模型下数据流自适应网格密度聚类算法，采用网格-密度的聚类算法，基于衰减窗口模型设计时态和空间的自适应密度阈值.文献[19]利用基于网格的聚类思想，将数据集合划分为一定数量的网格单元，并结合加权信息熵的概念，对 OPTICS 算法进行了优化，该算法在一定程度上降低了运行时间，具有较优的稳定性.

2 问题描述及预备知识

2.1 轨迹数据语义结构

位置获取技术的发展产生了大量的空间轨迹数据，这些轨迹代表了如人、车辆等在内的各种移动物体的运动行为.移动轨迹通常由一系列有序的点组成，如 $p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_n$，每个点都可以用一组空间坐标以及时间戳的三元组来表示，$p=(x,y,t)$.以下将给出本文使用的轨迹相关定义：

定义 2.1 GPS 轨迹.用户的 GPS 轨迹是一系列带时间戳的序列点，即

$$\mathrm{Traj}=\langle p_0,p_1,p_2,\cdots,p_k\rangle \tag{1}$$

式中 $p_i=(x_i,y_i,t_i)(i=0,1,\cdots,k)$；$t_i$ 是一个时间戳，(x_i,y_i)是点的二维坐标，包含着点 p_i 的纬度 p_i.lat 和经度 p_i.lon 信息.

定义 2.2 $\mathrm{Dis}(p_i,p_j)$.空间中两个 GPS 点之间的距离，该距离由欧几里得距离计算得到，即：

$$\mathrm{Dis}(p_i,p_j)=\sqrt{(x_i-x_j)^2+(y_i-y_j)^2} \tag{2}$$

2.2 分布式聚类

针对海量、分散的数据，分布式聚类算法以分布的数据资源为基础，独立地对局部数据进行聚类分析，而后将分析的结果反馈到主服务器等待整合.分布式聚类算法的核心思想分为四步[20]：

(Ⅰ)构建局部数据模型.

(Ⅱ)对每一部分进行并行聚类.

(Ⅲ)整合局部聚类结果以获得整体聚类结果.

(Ⅳ)根据全局聚类的结果对局部聚类的归属进行相应调整.

2.3 DBSCAN 聚类算法

DBSCAN 算法[15]是一种基于密度的空间聚类算法.该算法将密度足够大的区域划分为簇，并在具有噪声的空间数据库中发现任意形状的簇，它将簇定义为密度相连的点的最大集合.它要求包含在一定区域内对象的数目不小于给定的阈值.当遇到大规模数据时，由于时间复杂度增大而且需要消耗大量的内存，使用 DBSCAN 算法将不能有效地对数据进行聚类分析.

DBSCAN 算法的步骤为：输入扫描半径(Eps)和最小包含点数(minPts).任选一个数据集中未被访问过的对象，找出与其距离小于或等于 Eps 的所

有邻居对象.如果邻居对象的数量大于或等于minPts,则当前对象与其邻居对象形成一个簇,并且触发对象被标记为已访问,然后递归,以相同的方法处理该簇内所有未被标记为已访问的对象,从而对簇进行扩展;如果邻居对象的数量小于 minPts,则该对象暂时被标记作为噪声点;如果簇内的所有对象均被标记为已访问,那么就用同样的方法去处理数据集中其他未被访问的点对象.根据本文的应用场景,给出下面几个定义[21]:

定义 2.3 对象的密度.地理空间中任意一个对象的密度是以该对象为圆心,Eps 为半径的圆形区域内包含的对象的数目.

定义 2.4 对象的领域.空间中任一对象的领域是以该对象为圆心,Eps 为半径的圆形区域内包含的对象的集合,即

$$\mathrm{Eps}(p_i)=\{p_j \in X \mid \mathrm{Dis}(p_i,p_j) \leqslant \mathrm{Eps}\} \quad (3)$$

定义 2.5 核心对象.在所有对象构成的空间中,如果某一对象的密度大于给定的阈值 minPts,则为核心对象.

定义 2.6 对象间直接密度可达.对象 p_i 从对象 p_j 直接密度可达,如果满足以下条件:

(Ⅰ) p_i 处于 p_j 的领域中,即 $p_i \in \mathrm{Eps}(p_j)$;

(Ⅱ) p_j 是核心对象,即 $|\mathrm{Eps}(p_j)| \geqslant \min \mathrm{Pts}$.

3 基于轨迹数据密度分区的分布式并行聚类方法

针对规模庞大、时效性强、处理过程复杂的数学模型等轨迹数据的特点,本文提出了一种基于轨迹数据密度分区的分布式并行聚类方法.本方法基于DBSCAN 算法,通过分布式处理过程,有效地保证了轨迹信息分析在高速的环境下工作.

3.1 整体框架

本方法由两大部分组成:数据分区处理模块和集中分布式处理模块.整体执行过程如图 1 所示.首先提出了基于密度的分区算法,利用该算法将整个空间数据库分成若干矩形区域得到分区集合(partition set),从而构建局部聚类模型;然后提出了分布式并行 DBSCAN 算法,利用管理服务器(manager)将已划分好的分区集合转换成相对应的任务队列上传至网络中,工作服务器(worker)从网络中获取工作任务,而后 worker 对任务实行并行局部聚类,并将完成的任务传回至网络中,最后由 manager 对完成的任务进行合并处理并得到最终聚类结果,以此达到分布式聚类处理过程.

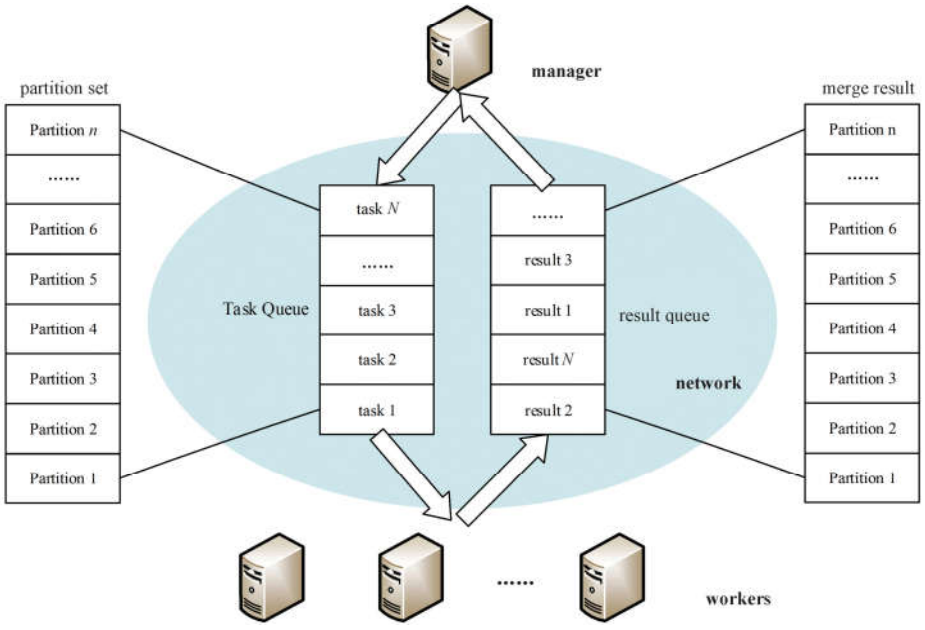

图 1 算法整体架构

Fig.1 The overall architecture of the algorithm

3.2 基于密度的分区算法

针对规模庞大的轨迹数据集,数据分区技术可以将整个聚类过程划分成多个任务进行并行聚类,有效缓解 CPU 紧张的局面,提高聚类分析速率.同时,合理的数据分区结果能够保证各任务的计算量相当.为此,本文首先提出了基于密度的分区算法

(density-based partitioning,DbP),将整个空间数据集放在一个规则的矩形内,根据其空间分布的密度特性,利用每次分割后矩形最长维度的改变,将整个轨迹数据集划分为规则的、密度量大致相同的若干局部矩形分区.

给定由大量轨迹坐标 $p_i=(p_i.\mathrm{lat},p_i.\mathrm{lon},t_i)(i=0,1,\cdots,k)$ 构成的数据集 X 以及参数 Eps.坐标集合 X 的边界为由其中点的最大经、纬度和最小经、纬度(max_Latitude,min_Latitude,max_Longitude,min_Longitude)组成的矩形.DbP 算法的核心思想为:沿着 X 最长维度将其拆分成两个分区,如果 X 的最长维度是纬度,那么分界线垂直于 X 的纬线,其纬度(bound_Latitude)为

(max_Latitude + min_Latitude)/2.

如果最长维度是经度,那么分界线垂直于 X 的经线,其经度(bound_Longitude)为

(max_Longitude + min_Longitude)/2.

这两个分区的每一分区将被继续拆分,依此类推.重复此过程直到满足以下任一条件:

(Ⅰ)该区域内的点数(point_cnt)小于或等于设置的最小点数(min_point_cnt);

(Ⅱ)每一个被拆分成分区的最短边的长度小于 2 倍的 Eps.其最短边可能是该分区的横向边(D_value_Latitude)或者是该分区的纵向边(D_value_Longitude),所对应的长度分别为

|max_Latitude − min_Latitude|,

|max_Longitude − min_Longitude|.

在分区结束后,可能会割断一些数据间本来存在的联系,会出现本属于一个类却被分到两个相邻分区的情况.当然,此种情况可利用局部聚类合并算法去解决,但是该算法必须对边界区域内的数据集重新查询它的邻域,增加了合并算法的复杂度,也使得聚类分析时间变长.观察可以发现,受影响的数据仅为部分分区边界上的数据集,因此在进行分区时,可将边界值稍作调整便可有效解决此问题,调整后的分界线的纬线和经线所对应的 bound_Latitude,bound_Longitude 分别为

(max_Latitude + min_Latitude)/2 + Eps,

(max_Longitude + min_Longitude)/2 + Eps.

DbP 算法分区过程如图 2 所示.最外层的矩形内包含了数据集 X 中所有的轨迹坐标数据,沿着该矩形的最长维度即纬线的中点作垂线,该垂线的纬度加上 Eps 为分区的分界线即图中虚线,该分界线将该区域分成了左、右两个子分区;先对左子分区进行分割,从图 2 可以看出,该分区的最长维度是经度,从而该左子分区的分界线为其经度加上 Eps,由此其被分成了上、下两个子分区;然后该上子分区又被分割为左右两个子区,此时我们发现该左子分区满足了加入分区集合的条件,那么就将该区域加入分区集合并开始对右子分区进行分割;以此类推直到得到所有分区.同时,图 2 中显示的数字代表了该分割区域加入分区集合的顺序.

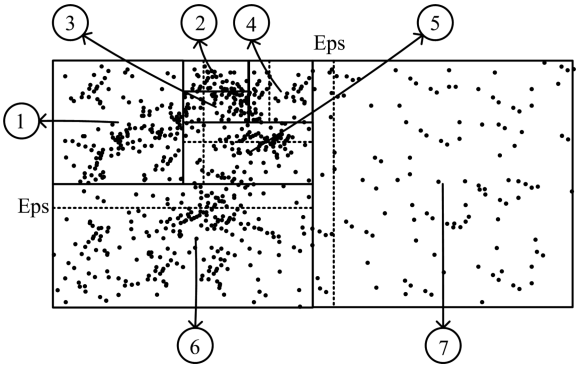

图 2　基于密度的分区算法具体分区过程

Fig.2　The partition process of DbP

DbP 算法对 X 数据集进行分区的过程是一个前序遍历的过程,其对应的前序遍历树如图 3 所示.图 3 中标号的数字与图 2 中的数字相对应,代表了区域加入分区集合的顺序.树的层数代表了分割区域的次数.通过第一次分割将整个数据集分成左右两个子区,先遍历左子区(上子区)部分,直到该区域满足加入分区集合的条件,再返回根节点并继续遍历右子区(下子区),直到得到所有分区整个遍历过程结束.DbP 算法具体实现过程在算法 3.1 给出.

算法 3.1　基于密度的分区算法

输入:数据集 X,Eps,最小点数 min_point_cnt;

输出:分区集合 partitions.

(1)partitions = [] /* 构建分区集合 */

(2)def partitioning(X,min_point_cnt, Eps):/* 定义分区函数 */

(3)if point_cnt <= min_point_cnt:/* 判断当前区域内点数是否小于指定的最小分区点数量 */

(4)partitions.append(X)　　/* 将当前区域加入分区集合 */

(5)end if

(6)if D_value_Latitude<=(2 * Eps): /* 判断纬度的差值是否小于等于 2 倍 Eps */

(7)partitions.append(X)

(8)end if

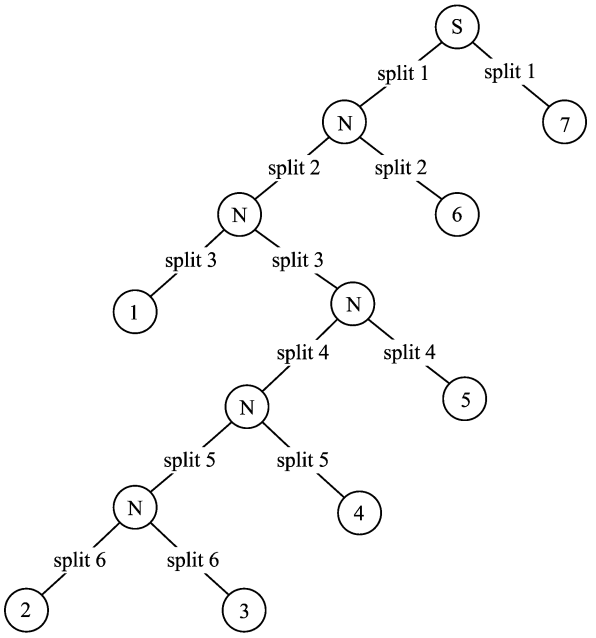

图 3 基于密度的分区算法前序遍历树

Fig.3 The pre-travel tree of DbP

(9)if D_value_Longitude<=(2 * Eps): /* 判断经度的差值是否小于等于 2 倍 Eps */

(10)partitions.append(X)

(11)endif

(12)if D_value_Latitude>D_value_Longitude: /* 判断纬度的差值是否大于经度的差值 */

(13)X_left = X.copy() /* 将当前集合复制一份用以分割出左子分区 */

(14)X_left=X_left[min_Latitude <= Latitude < (bound_Latitude+Eps)] /* 计算左子分区 */

(15)partitioning (X_left, min_point_cnt, Eps) /* 迭代左子分区 */

(16)X_right = X.copy() /* 将当前集合复制一份用以分割出右子分区 */

(17)X_right=X_right[(bound_Latitude-Eps)<=Latitude<=max_Latitude] /* 计算右子分区 */

(18)partitioning(X_right,min_point_cnt,Eps) /* 迭代右子分区 */

(19)else: /* 纬度的差值小于等于经度的差值 */

(20)X_up=X.copy()/* 将当前集合复制一份用以分割出上子分区 */

(21)X_up=X_up[min_Longitude<=Longitude<(bound_Longitude+Eps)] /* 计算左子分区 */

(22)partitioning (X_up, min_point_cnt, Eps) /* 迭代左子分区 */

(23)X_down=X.copy()/* 将当前集合复制一份用以分割出下子分区 */

(24)X_down=X_down[(bound_Longitude-Eps)<=Longitude<=max_Longitude]/* 计算下子分区 */

(25)partitioning (X_down, min_point_cnt, Eps) /* 迭代下子分区 */

3.3 分布式并行 DBSCAN 算法

DBSCAN 算法能够有效地对空间数据进行聚类分析,然而当数据量过大时,传统的串行聚类算法将面临时延大、CPU 占用过高、内存占用率大的问题.为此,本文提出了分布式并行 DBSCAN 算法(distributed parallel dbscan, DPDBSCAN),基于 DbP 算法得到的局部模型对其实现分布式并行聚类,并进行整合得到最终聚类结果.

DPDBSCAN 算法包含两个子算法,分别运行在管理服务器(manager)端和工作服务器(worker)端 .manager 首先建立任务和结果队列,然后调用 DbP 算法,将得到的分区集合加入到任务队列中,之后 manager 将一直处于等待状态直到结果队列为满,最后将结果队列中的局部聚类结果进行合并,并输出整个聚类结果,管理服务器的算法将在算法 3.2 给出.worker 从网络中获取任务队列,随后每个 worker 将依次对任务进行局部 DBSCAN 聚类并将结果返回到结果队列中,工作服务器的算法在算法 3.3 给出.

算法 3.2 分布式并行 DBSCAN 算法(manager)

输入:数据集 X,Eps,min_point_cnt;

输出:聚类结果集合 resultData.

1)task_queue = Queue.Queue()
/* 创建任务队列 */

2)result_queue = Queue.Queue()
/* 创建结果队列 */

3)QueueManager.register('get_task_queue')
/* 网络中注册任务队列 */

4)QueueManager.register('get_result_queue')
/* 网络中注册结果队列 */

5)task = manager.get_task_queue()
/* 获取网络中的任务队列 */

6)result = manager.get_result_queue()/* 获取网络中的结果队列 */

7)partitions = partitioning(X, min_point_cnt, Eps)
/* 执行分区算法并将结果赋值给分区集合 */

8)for split in partitions: /* 循环将分区集合的分区放置任务队列 */

9) task.put(split)

10)end for

11)resultData = [] /* 定义任务结果集合 */

```
12)while(true):/* 循环等待,抽取结果队列的数据 */
13)        if(result ! = null):
/*如果结果队列不为空,则将结果队列中的数据追加到结果集合中 */
14)            resultData.append(result.get())
15)if(resultData.len = splits.len)
/* 如果结果集合中的个数等于分区个数则表示任务执行完成 */
16)            return resultData
```

算法 3.3　分布式并行 DBSCAN 算法(worker)

```
输入:分区 split,Eps;
输出:聚类结果队列 result.
1)QueueManager.register('get_task_queue')
                                /* 注册网络任务队列 */
2)QueueManager.register('get_result_queue')
                                /*注册网络结果队列 */
3)task = m.get_task_queue()
                                /*获取网络任务队列 */
4)result = m.get_result_queue()
                                /*获取网络结果队列 */
5)while True: /* 循环获取任务队列数据 */
6)if task.isEmpty:
                        /* 如果任务队列无数据则退出 */
7)     return
8)   else:
9)        split= task.get()
                            /*从任务队列中获取分区 */
10)        r = dbscan_model(split, Eps)
                                /* 执行 DBSCAN 算法 */
11)     result.put(r)    /*将 DBSCAN 算法得到的聚类结果放置结果队列中*/
```

DBSCAN 聚类算法的时间复杂度为 $O(n^2)$，其中 n 为数据集中对象的个数，当 n 的规模很大时，其算法耗时为 n^2t .本文提出的方法的执行时间主要分布在局部聚类分析的时间及数据传输上，分区及局部结果入队的时间对总时间影响不大.设 k 为分区的总个数，由于本文是基于密度的分区，保证了每个区域内的密度大致相同，因此，每个区域内的对象个数为 n/k ；设传输一个对象局部聚类分析的结果的时间为 T_{eq} ，所以方法总耗时为

$$T=\frac{n^2t}{(n/k)^2t+nT_{eq}/k} \tag{4}$$

由此可见，选取一个合理的分区数目将有效地提高本方法的运算速率.

4　实验与分析

4.1　实验环境

本文算法的实验环境是由 6 台高性能服务器搭建的集群，其中包含一个管理服务器(manager)以及五个工作服务器(worker).集群中每个节点的具体参数如表 1 所示.

表 1　实验服务器节点参数

Tab.1　Configuration of the node

操作系统	Linux 操作系统
CPU	8 核(使用 1 核)，2599.934MHz
内存	128G
编程语言	Python

4.2　数据集

本实验采用的轨迹数据来自于微软亚洲研究院 GeoLife[22-24] 的一个真实轨迹数据集.该 GPS 轨迹数据集由 182 个用户在 2007 年 4 月至 2012 年 8 月期间内收集完成.此数据集共包含 17 621 条轨迹，这些轨迹由一系列带时间戳的点组成，每个点的信息包括纬度、经度和高度.该 GPS 轨迹数据集展现了大范围的用户户外移动情况，不仅包含了用户回家、上班等生活轨迹，也包含了诸如购物、观光等娱乐和体育活动.数据集覆盖的范围遍布中国的 30 多个省市、美国及欧洲部分城市，本文从这些数据中提取出北京地区的用户轨迹数据进行研究.

4.3　评价指标

加速比.同一个任务在单处理器系统和并行处理器系统中运行消耗的时间的比率，定义为

$$S_p=T_1/T_p \tag{5}$$

式中，T_1 表示单机模式下的运行时间，T_p 表示在有 p 个处理器并行系统中的运行时间.

针对本文的应用场景，采用空间索引，DBSCAN 算法的时间复杂度为 $O(n\log n)$，那么加速比为

$$S_p=\frac{n\log n}{nk(\log n-\log k)+nT_{eq}/t} \tag{6}$$

4.4　实验结果与分析

我们主要对 DbP 和 DPDBSCAN 两个算法进行评价分析来验证本文提出的基于轨迹数据密度分区的分布式并行聚类方法的可行性和高效性.

4.4.1　基于密度的分区算法评价分析

在这部分的分析中，第 1 个实验记录了本方法

中各个阶段的执行时间，如图 4 所示，包含了分区时间，局部聚类结果入队时间以及聚类时间.随着数据量的增多，分区时间及入队时间并没有明显的变化，其过程对整个方法的执行效率影响不大，所以，本文提出的方法是合理的，算法运行时间主要集中于聚类过程，验证了之前的理论分析；同时，基于密度的分区算法可以有效、高速地对数据集进行分区，无论数据量增多多少，其性能不会受很大影响，为分布式并行聚类过程做了很好的基础工作.

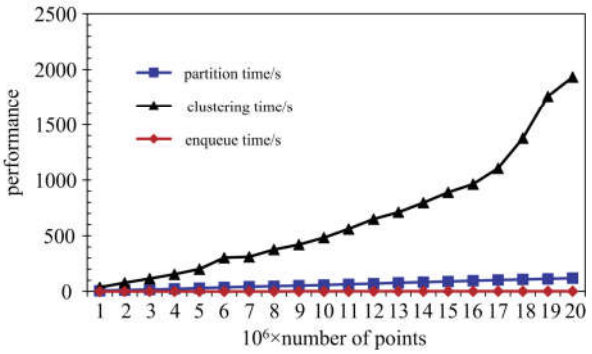

图 4 各阶段的执行时间

Fig.4 The execution time of each stage

第 2 个实验分析了网络传输时间与分区数量的关系，从图 5 可以看出，分区的数量会影响网络传输的总体时间.分区数量较少时，大量的时间在聚类计算，消耗在数据传输的时间就较少；当分区数量过大时，网络中需要传输大量的小块分区数量，所以消耗在网络上的时间就会很多.当分区数量更大时，计算节点计算分区的时间与传输数据的时间达到了一定的平衡，此时网络消耗的时间又会减少.综上所述，需要找到一个较好的分区数量来进行分布式计算.

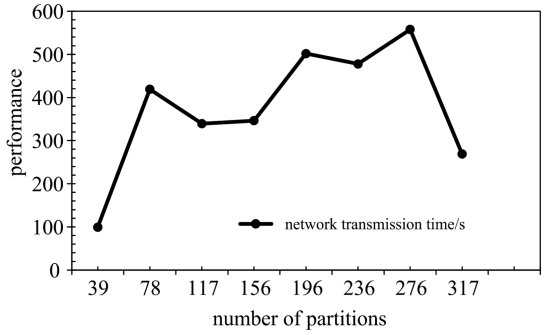

图 5 分区数量与网络传输时间的关系

Fig.5 The relationship between number of partition and network transmission time

第 3 个实验分析了分区时间与分区数量之间的关系，从图 6 可以看出，随着分区数量的持续增加，分区所花费的时间在逐渐增多，但即使增多，所消耗的时间也是相对较少的.所以在考虑总体时间的时候可以忽略分区数量对分区时间的影响.

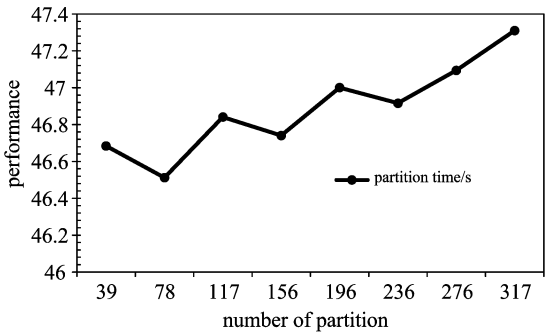

图 6 分区时间与分区数量的关系

Fig.6 The relationship between number of partition and partition time

4.4.2 分布式并行 DBSCAN 算法评价分析

在这一部分的评价分析中，第 1 个实验对比了传统的 DBSCAN 算法的执行时间与本文提出的方法的执行时间，如图 7 所示，从图 7 可以看出，在数据量相同的情况下，本文所提出的方法执行时间明显低于传统的串行聚类方法，同时，随着数据量的增多，本文方法的计算速率较传统的 DBSCAN 算法具有更加明显的优势.

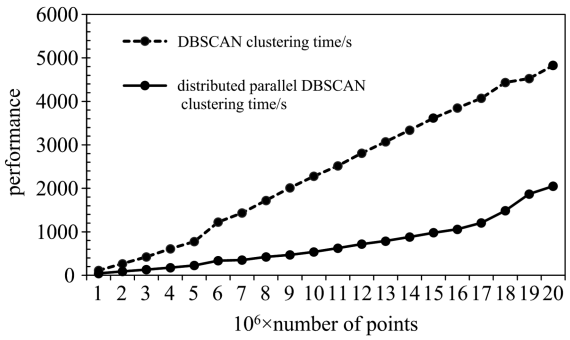

图 7 DBSCAN 算法与本文提出的方法的性能比较

Fig.7 Performance comparison on DBSCAN and our method

第 2 个实验如图 8 所示.图 8 展现了当分区数量基本恒定的情况下，随着数据量的变化其加速比的变化情况.当数据量为 900 万时，加速比呈现最高状态，可达到近 4.3；然而，当数据量以 900 万为分界线逐步减少或增多时，其加速比会呈现逐步降低的趋势，所以，针对不同的输入数据量，应综合考虑数据量与分区数量的关系，以获得最佳的加速比结果.

第 3 个实验如图 9 所示.图 9 对比分析了本文方法与 POPTICS[25] 方法的性能.POPTICS 是一种分布式 OPTICS 聚类方法，可以根据不相交的数据结构和 Prim 的最小生成树(MST)同时聚类一组点

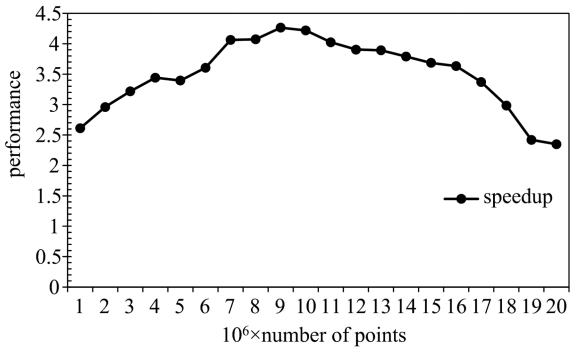

图 8　不同数据量下加速比的变化

Fig.8　The speedup in different number of points

数据。该方法使用多个 CPU 核或服务器节点实现了基于密度聚类方法的分布式处理过程.图 9 显示了不同数据量的情况下两种方法的加速比结果,数据量在 16 000 000 以下时,我们所提出的方法性能优于 POPTICS 方法,数据量为 10 000 000 时结果最优,然而随着数据量的增多性能有下降趋势,所以应综合考虑本方法中分区量等参数的关系以提高可扩展性.

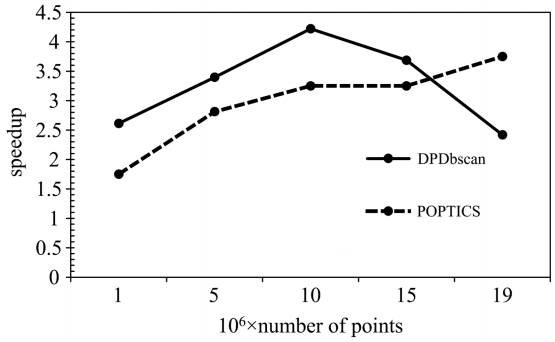

图 9　POPTICS 方法与本文提出的方法的性能比较

Fig.9　Performance comparison on POPTICS and our method

5　结论

本文提出了一种基于轨迹数据密度分区的分布式并行聚类方法.根据轨迹数据的空间分布密度对整个数据集进行合理划分,然后转换成相应的任务进行局部并行聚类.本算法能够有效缓解串行聚类算法因数据规模过大内存需求和 CPU 紧张的局面.实验结果表明,本文提出的方法在有 5 台工作服务器的情况下,加速比最高可达到近 4.3.

从实验结果可以发现,数据量和分区数量的变化都将影响加速比的结果,因此如何选取参数得到最优的加速比结果将是继续研究的内容;同时,本文中的聚类分析只考虑了 2 维的情况,实际应用中很多都还存在多维的情况,未来还需展开对多维数据的分布式聚类方法的研究.

参考文献(References)

[1] FANG Z X, SHAW S L, TU W, et al. Spatiotemporal analysis of critical transportation links based on time geographic concepts: a case study of critical bridges in Wuhan, China[J]. Journal of Transport Geography, 2012, 23(3): 44-59.

[2] LI Q N, ZHENG Y, XIE X, et al. Mining user similarity based on location history[C]// Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems. Irvine, USA: ACM Press, 2008: No. 34.

[3] ZHENG Y, ZHANG L Z, MA Z M, et al. Recommending friends and locations based on individual location history[J]. ACM Transactions on the Web, 2011, 5(1): 5(1-44).

[4] REHM F. Clustering of Flight Tracks[M]//AIAA Infotech@ Aerospace 2010. 2010: 3412.

[5] 赵恩来, 郝文宁, 赵飞, 等. 改进的基于密度的航迹聚类算法[J]. 计算机工程, 2011, 37(9): 270-272.
ZHAO Enlai, HAO Wenning, ZHAO Fei, et al. Improved track clustering algorithm based on density [J]. Computer Engineering, 2011, 37(9): 270-272.

[6] YUAN G, XIA S, ZHANG L, et al. An efficient trajectory-clustering algorithm based on an index tree [J]. Transactions of the Institute of Measurement and Control, 2012, 34(7): 850-861.

[7] BERMINGHAM L, LEE I. A general methodology for n-dimensional trajectory clustering[J]. Expert Systems with Applications, 2015, 42(21): 7573-7581.

[8] LEE J G, HAN J, WHANG K Y. Trajectory clustering: A partition-and-group framework [C]// Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data. Beijing: ACM Press, 2007: 593-604.

[9] I ZAKIAN Z, MESGARI M S, ABRAHAM A. Automated clustering of trajectory data using a particle swarm optimization[J]. Computers, Environment and Urban Systems, 2016, 55: 55-65.

[10] AGGARWal C C, Li Y, Wang J Y, et al. Frequent pattern mining with uncertain data[C]// Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Pairs: ACM Press, 2009: 29-38.

[11] GUPTA A, HARINARAYAN V, QUASS D. Aggregate-query processing in data warehousing environments [C]// Proceedings of the 21th International Conference on Very Large Data Bases.

San Francisco: ACM Press, 1995: 358-369.

[12] HARTIGAN J A, WONG M A. Algorithm AS 136: A k-means clustering algorithm[J]. Journal of the Royal Statistical Society, Series C (Applied Statistics), 1979, 28(1): 100-108.

[13] GUHA S, RASTOGI R, SHIM K. CURE: An efficient clustering algorithm for large databases[J]. ACM SIGMOD Record, 1998, 27(2): 73-84.

[14] ZHANG T, RAMAKRISHNAN R, LIVNY M. BIRCH: A new data clustering algorithm and its applications [J]. Data Mining and Knowledge Discovery, 1997, 1(2): 141-182.

[15] ESTER M, KRIEGEL H P, SANDER J, et al. A density-based algorithm for discovering clusters in large spatial databases with noise[C]// Proceedings of the International Conference on Knowledge Discovery and Data Mining. Portland: ACM Press, 1996: 226-231.

[16] KUMAR K M, REDDY A R M. A fast DBSCAN clustering algorithm by accelerating neighbor searching using Groups method[J]. Pattern Recognition, 2016, 58: 39-48.

[17] SMITI A, ELOUDI Z. Soft DBSCAN: Improving DBSCAN clustering method using fuzzy set theory [C]// The 6th International Conference on Human System Interaction. INSPEC, 2013: 380-385.

[18] 刘卓，杨悦，张健沛，等. 不确定度模型下数据流自适应网格密度聚类算法[J]. 计算机研究与发展，2014，51(11)：2518-2527.
LIU Zhuo, YANG Yue, ZHANG Jianpei, et al. An adaptive grid-density based data stream clustering algorithm based on uncertainty model[J]. Journal of Computer Research and Development, 2014, 51(11): 2518-2527.

[19] 安建瑞，张龙波，王雷，等. 一种基于网格与加权信息熵的OPTICS改进算法[J]. 计算机工程，2017，43(2)：206-209.
AN Jianrui, ZHANG Longbo, WANG Lei et al. An improved OPTICS algorithm based on grid and weighted information entropy [J]. Computer Engineering, 2017, 43(2): 206-209.

[20] 倪巍伟，陈耿，吴英杰，等. 一种基于局部密度的分布式聚类挖掘算法[J]. 软件学报，2008，19(9)：2339-2348.
NI Weiwei, CHEN Geng, WU Yingjie, et al. Local density based distributed clustering algorithm [J]. Journal of Software, 2008, 19(9), 2339-2348.

[21] TRAN T N, DRAB K, DASZYKOWSKI M. Revised DBSCAN algorithm to cluster data with dense adjacent clusters[J]. Chemometrics and Intelligent Laboratory Systems, 2013, 120(2): 92-96.

[22] ZHENG Yu, ZHANG Lizhu, XIE Xing, et al. Mining interesting locations and travel sequences from GPS trajectories [C]// Proceedings of International Conference on World Wild Web. Madrid, Spain: ACM Press: 791-800.

[23] ZHENG Yu, LI Quannan, CHEN Yukun, et al. Understanding mobility based on GPS data [C]// Proceedings of ACM Conference on Ubiquitous Computing . Seoul, Korea: ACM Press, 2008: 312-321.

[24] ZHENG Yu, XIE Xing, MA Weiying, GeoLife: A collaborative social networking service among user, location and trajectory[J]. Bulletin of the Technical Committee on Data Engineering, 2010, 33(2): 32-40.

[25] PATWARY M A, PALSETIA D, AGRAWAL A, et al. Scalable parallel OPTICS data clustering using graph algorithmic techniques[C]//Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. Denver: ACM Press, 2013: No.49(1-12).