计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.190700071

一种改进的 DBSCAN 算法在 Spark 平台上的应用

邓定胜
四川民族学院理工学院　四川 康定 626001

摘　要　针对 DBSCAN(Density-Based Spatial Clustering of Applications with Noise)聚类算法内存占用率较高的问题,文中将改进的 DBSCAN 聚类算法与 Spark 平台并行聚类计算理论相结合,对海量数据采用分而治之的办法进行聚类处理,大幅减小了算法对内存的占用率。实验仿真结果表明,所提出的并行计算方法能够有效缓解内存不足的问题,并且该方法也能够用来评价 DBSCAN 聚类算法在 Hadoop 平台下的聚类分析效果,还能对两种聚类方法进行对比分析,从而获得较好的计算性能;且其比在 Hadoop 平台上的计算加速度提高了 24%左右,因此可以用以评价 DBSCAN 聚类算法在聚类处理方面的优劣。

关键词:并行计算;DBSCAN;聚类算法;Spark;聚类加速比
中图法分类号　TP391

Application of Improved DBSCAN Algorithm on Spark Platform

DENG Ding-sheng
School of Science and Technology, Sichuan Minzu College, Kangding, Sichuan 626001, China

Abstract　Aiming at the problem of high memory occupancy of DBSCAN(Density-Based Spatial Clustering of Applications with Noise) clustering algorithm, this paper combines the improved DBSCAN clustering algorithm with the parallel clustering calculation theory of Spark platform, and the clustering and processing methods for massive data are clustered, which greatly reduces the memory usage of the algorithm. The experimental simulation results show that the proposed parallel computing method can effectively reduce the shortage of memory, and it also can be used to evaluate the clustering effect of the DBSCAN clustering algorithm on the Hadoop platform, and compare and analyze the two clustering methods to obtain better computing performance. Besides, the acceleration is increased by about 24% compared with that on the Hadoop platform. The proposed method can be used to evaluate the pros and cons of the DBSCAN clustering algorithm in clustering.

Keywords　Parallel computing, DBSCAN, Clustering algorithm, Spark, Clustering acceleration ratio

随着科技的快速发展,依靠大数据来进行计算的方式越来越普遍,云计算服务也应运而生。目前,相关领域的学者急需解决的关键技术是在大量繁杂的数据中抓取到有效的数据。国内外学者已经研发出了挖掘海量数据的新技术、新方法,例如划分大数据、数据重新聚类等,并将其成功应用到其所从事的行业[1];文献[2]采用自适应的 Eps 参数使得 DBSCAN 算法对具有不同密度的簇的数据集进行聚类;文献[3]提出了一种利用新的模糊聚类有效性指标来判断最佳聚类数,用以消除密度聚类算法对参数的敏感性,在对数据进行聚类时,聚类质量较原始密度聚类算法在准确性和自适应性方面都显示了较好的效果。数据类型关联理论对数据挖掘结果具有较大的影响,在众多的数据关联理论中,DBSCAN 算法具有较为重要的作用以及较好地数据关联效果[4]。文献[5]指出 DBSCAN 关联算法是依靠数据密度性质进行聚类关联的一种算法。DBSCAN 算法的核心理论是:首先要求在数据组中关键点周围的设定范围内应包含一定数量的数据点,然后对每一个范围进行判断,最后对密度较高的范围进行归类。DBSCAN 算法能够较好地适应数据组内部存在的噪声的影响[6];而且,算法的聚类分析过程较短,能够较好地满足大幅增加数据聚类时的计算要求[7]。但是,DBSCAN 算法亦有不可避免的缺陷,因 DBSCAN 算法在对数据进行聚类分析时会对整个数据组进行全局密度划分,这就导致了 DBSCAN 算法存在如下缺点:1)当需要数据挖掘的数据组数量较大时,需要增大计算机内存来满足 DBSCAN 算法聚类的精确度[8];2)在某些较为特殊的数据组内,若数据点之间的密度存在较大的差异,则会导致 DBSCAN 算法的聚类精度不高[9]。基于以上不足,本文采用先分再聚、并行计算的方法对 DBSCAN 算法进行改进[10],并基于一系列的降低维数以及增加数据量的方法[11-12],改善了聚类分析的计算效果。最后,将改进后的 DBSCAN 算法在 Spark 平台上进行了实际应用。

基金项目:四川民族学院自然科学重点项目(XYZB19001ZA);四川省教育厅自然科学重点项目(17ZA0295);四川民族学院 2017 年应用型示范课程项目(sfkc201705);国家自然科学基金项目(11461058)
This work was supported by the Key Project of Natural Science of Sichuan Minzu College(XYZB19001ZA), Key Project of Natural Science of Sichuan Provincial Education Department (17ZA0295), 2017 Applied Demonstration Course Project of Sichuan Minzu College (sfkc201705) and National Natural Science Foundation of China(11461058).
通信作者:邓定胜(307729837@qq.com)

1 传统 DBSCAN 算法的不足

传统的 DBSCAN 聚类算法主要依靠数据组内部数据点之间的密度特性来进行聚类计算[13]。该算法能够降低数据内部的噪声影响，同时能够对各种形式以及大小各异的数据组进行聚类分析[14]。DBSCAN 聚类算法能够避免聚类过程中漏掉许多数据的缺点[15]，但当数据组的簇密度分布波动较大时，聚类分析算法就会产生一定的缺陷，特别是对一些特殊的高维度数据数组内部的数据点密度进行聚类分析时就显得相对困难[16]。

另外，DBSCAN 聚类算法对输入参数极为敏感，如果参数值选择不当，就会导致 DBSCAN 聚类算法的结果出现偏差；同时，DBSCAN 聚类算法对数组进行全局密度划分时，极易出现数据点分布不均匀的现象，从而导致 DBSCAN 算法聚类分析结果不理想。虽然针对这些客观存在的问题，国内外学者也进行了大量的研究，并给出了各种解决方案[17]，但已有研究并未较好地解决这个问题。例如，DBSCAN 聚类算法时常会在获得一个局部最优值时终结，并且只适合于数值型数据的聚类，还只能发现聚类结果为凸形的数据集，针对该问题，国外的 Kaufman 提出了 K 中心的算法 PAM；国内的 Xu 对于 DBSCAN 聚类算法中有关随机选择初始聚类中心的不足，提出了一种新的基于数据样本分布选取初始聚类中心的算法，增大了 K-Means 聚类算法的准确度。

2 改进的 DBSCAN 聚类算法模型

DBSCAN 聚类理论的具体原理可参考文献[18]，其涉及的相关概念和公式的描述如下。

(1)eps 邻域：以待处理数据集 D 中的数据对象 p 为圆心，eps 为半径的区域。其中 eps 邻域内包含的点的集合如下：

$$N_{eps}(p)=\{q\in D/dist(p,q)\leqslant eps\} \tag{1}$$

其中，$N_{eps}(p)$ 表示 p 的 ε-邻域的样本集合，$dist(p,q)$ 表示样本 p 与样本 q 的欧氏距离。

(2)核心点：对于数据集 D 中的点 p，核心点在 eps 邻域内的密度值 Num 满足如下的表达式：

$$Num\geqslant MinPts \tag{2}$$

(3)边界点：对于数据集 D 中的点 p，边界点在 eps 邻域内的密度值 Num 满足如下的表达式：

$$Num\leqslant MinPts \tag{3}$$

(4)直接密度可达：给定对象 p 和 q，其中 p 为核心点，且 p 的 eps 邻域包含 q，称 q 到 p 为直接密度可达，其表达式为：

$$q\in N_{eps}(p),|N_{eps}(p)|\geqslant MinPts \tag{4}$$

通过以上基本概念可得 DBSCAN 的主要聚类技术路线如下：

(1)对于数组聚类后的任一数据点，定义其密度的半径至少应大于数组中数据的最小数量，DBSCAN 聚类算法主要基于阈值以及 $MinPts$ 的改变来组成数组内部的控制簇[19]。

(2)在对数组内部的数据点进行聚类之前，应先建立一个聚类分类的数组集合 D，从 D 中选取一个任意聚类对象，以 P 为原点，在设定好的半径内寻找到一个对象，这些对象均应采用 eps 半径、最小参数对象 $MinPts$。

(3)假定核心聚类对象为集合 P，其包含的数组数量应大于 $MinPts$，因此经过 DBSCAN 聚类算法能够确定一个客观合理的数据集合 P。

(4)最后，DBSCAN 聚类算法能够在这个区间内周而复始地找寻数据集合 P 中直接或间接的数据对象，直到聚类过程搜索不到新的数据点，此时聚类过程终止。

综合以上对 DBSCAN 聚类算法的改进，将任意两个特征数据点归纳到同一个数据簇中，此特征点周围的一些边界点亦能够被添加进这个相同的簇中，在聚类过程中，DBSCAN 聚类算法能够去除影响聚类结果的噪声数据点。

3 Spark 框架平台

Spark 成为各行业对大量数据进行聚类分析的重要平台，例如，伯克利大学曾以 Spark 平台为核心，将其整合成大数据聚类分析系统，并在此基础上提供了更高级的聚类计算模型[20]。

Spark 平台是伯克利大学专为大规模数据处理而设计的，被称为伯克利数据分析栈(BDAS)系统的关键部分，它承担着整个系统的所有程序编写操作。Spark 平台不仅包括 MapReduce (MR)算子以及 reduce 关系模型，还包括其他优秀的计算程序，如 filter，join 和 groupByKey 计算程序等。Spark 平台通过将大数据进行聚类划分，形成一种分布式的数据集合(RDD)，从而实现了聚类计算调控、数据集合排列以及数据压缩等综合操作，还能够对系统内部各个组建提供 API 接口等服务。并且系统下层部件能够基于 S 函数(Scala)进行程序的编写，其接口形式也与 S 函数的编程操作类似。Spark 平台的聚类计算流程如图 1 所示。

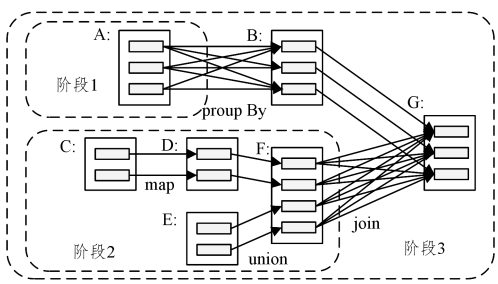


图 1 Spark 平台的聚类计算流程图

Fig. 1 Clustering calculation flow chart on Spark platform

为了在 Spark 平台上实现 DBSCAN 聚类算法引入了成熟的矩阵数据组概念，仅需对聚类数组进行较少次数的数据扫描。Spark 平台与 DBSCAN 聚类算法的结合应用，能够极大地提高聚类算法的关联规则，改善了大数据挖掘的聚类效率。在算法应用中首先要进行数据空间划分，保证划分后每个空间中的数据量大致相等。数据空间划分的具体步骤如算法 1 所示。

算法 1 数据空间划分

输入：原始数据集 D，最小半径 eps，每个划分空间最少包含点数 Points

输出：最终的划分空间集 P

1. 确定划分空间的数目：设总数据量是 M，则划分空间的数目 N 满足如下表达式：

$$N=M/Points \tag{5}$$

2. 确定分割数 $K_i(i=1,2,\cdots,n)$：n 为数据维数，K_i 的表达式如下。

$$K_i \approx \sqrt[n]{N} \&\& N = \prod_{i=1}^{n} K_i \tag{6}$$

3. 确定划分点:原始数据集的划分空间集为 P_0,划分后得到空间集为 P_i,具体如下。

3.1. 原始的空间集为:

$$P_0 = D \tag{7}$$

3.2. P_i 集合内的任意划分空间 p 为:

$$\forall p \in P_i \tag{8}$$

3.3. 设 p 在第 i 维的最小边界值为 p_{min},最大边界值为 p_{max},p 内数据点的数量为 M_p,划分 s 个小区间,则 s 满足如下表达式:

$$s = \left[\frac{p_{max} - p_{min}}{2 \cdot eps} \right] \tag{9}$$

3.4. 假设 j 为 p 上第 j 个小区间,则 j 可以表示为:

$$j \in \{1,2,\cdots,s\} \text{ to } s \tag{10}$$

3.5. 遍历区间(start,end)的数据量满足如下表达式:

$$\sum_{j=start}^{end} count \geqslant \frac{M_p}{K_i} \tag{11}$$

3.6. 将新的空间加入划分空间,表示为 P_i,并令:

$$start = end + 1 \tag{12}$$

3.7. 重复步骤 3.2—步骤 3.4,直到最后空间划分完毕。

对于数据划分的空间,执行本地 DBSCAN 算法,在所建立的矩阵中,行向量代表事物的集合,列向量代表事项的集合。行、列向量在矩阵中以 0/1 两个数字进行表示,这样能够大大减小聚类计算过程中的内存占用量,切实降低数组扫描的循环次数。基于 Spark 平台的 RDD 算法的工作流程如下:

(1)对所建的事物矩阵进行扫描,得到"1"集合 L_1;

(2)将 RDD 算法进行均分,形成 n 个数据子集合,将子集合的数据全部划分到各个工作节点上;

(3)建立局部计算矩阵,假设 $D_i(1 \leqslant i \leqslant n)$ 内部的事物数量为 J,L_1 的内部个数为 H,因此 L_1 与 D_i 组成了矩阵 $\boldsymbol{G}_i$,其表达式如下:

$$G_i = H \times (J+2) \tag{13}$$

其中,如果 H 与 L_1 中的项矩阵一一对应,那么就将矩阵中的相应位置设定为"1",反之设定为"0"。矩阵向量设定值以及转换如图 2 所示。

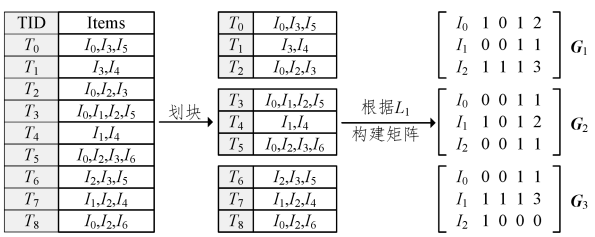


图 2 矩阵向量的转换

Fig. 2 Matrix vector transformation

4 聚类算法的并行化实现

搜集的大数据经过 DBSCAN 聚类算法进行类别划分后,形成不同的数据块,且每一个数据块均能够独自完成聚类计算。虽然 DBSCAN 聚类并行计算理论能够极大地增大计算效率,但是,从数据块边界层是否重复的角度出发,DBSCAN 聚类算法的数据块并不能够无限大。要提高 DBSCAN 聚类算法的运行效率,必须降低其自身的复杂度。基于 DBSCAN 聚类理论的计算流程,消耗内存的两个主要方面在于:

(1)数组内所有数据点的扩展过程。

(2)扩展过程中采用并行化或改进的 DBSCAN 聚类算法进行计算,对各个数据块的划分主要基于粗聚类以及粗聚类簇的并行计算,能够大大降低算法的复杂度,其计算流程如图 3、图 4 所示。

第一阶段:先对数据块进行粗聚类计算。在数据组中选出任一 q_1 点,并判断其是否为核心点;若 q_1 点能结合 eps 相邻的密度区域内的所有点构成粗聚类簇,则 q_1 点被定义为簇组中的核心点。然后再选取任一 q_2 点,对 q_2 点进行如 q_1 点的操作,重复此过程,直至没有簇组核心点出现为止,如图 3 所示。

第二阶段:对粗聚类簇组进行合并,其计算流程如图 4 所示。计算过程中,假定 a 和 b 点分别为 A 和 B 簇的核心点,此两点应满足如下规则。

1)当 a 点被判定为簇组核心点时,若 b 点也是核心点的话,则 a 和 b 点之间的距离应不大于 3eps。反之,若 b 点不是核心点,则对应的距离应小于 2eps。

2)当 a 点不是核心点时,若 b 点是核心数据点,那么 a 与 b 点之间的距离必须要不大于 2eps。若 b 点同样不是核心点,那 a 和 b 点的距离应不大于 1eps。

在只扩展 A 簇或 B 簇时,先判定图 3 中 a 与 b 点之间的数据点密度是否合适,若密度符合要求,那么对这两簇进行并行计算。为了确保算法能够判断数据点是否为真正的核心点,需将上述所有判断步骤均加入到算法的计算过程中,此时不需要对其进行展开计算,大大减少了聚类算法的计算量。并且 A 簇与 B 簇一旦进行了合并计算,其后面的数据点均不必进行展开计算,这大大降低了计算的复杂度。

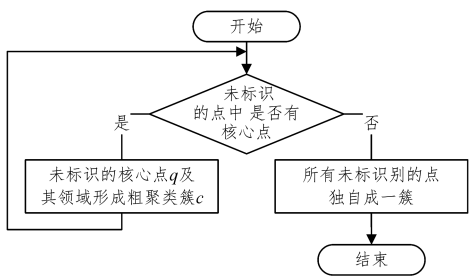


图 3 第一阶段粗聚类流程图

Fig. 3 Flow chart of first-stage rough clustering

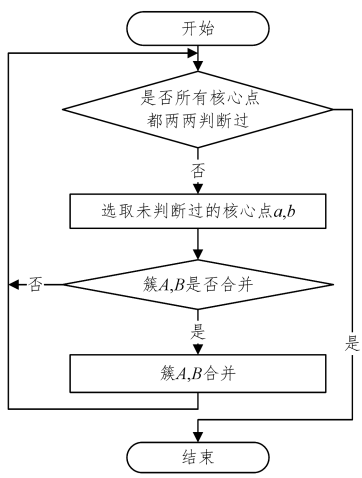
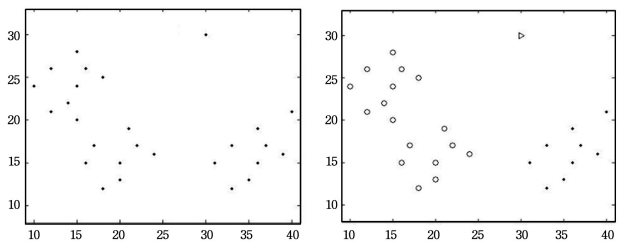


图 4 第二阶段粗聚类簇合并流程图

Fig. 4 Flow chart of second stage rough clustering cluster

图 5 给出了以上每一个阶段的计算结果，数据组经过第一阶段的聚类计算后，全部数据点都会被送入下一个阶段进行簇组粗聚类。聚类算法对所有数据组进行全局搜索后，应判断 a 点是否为核心数据点，如上所述循环往复。而 b 点也应完成 a 点的计算过程，这决定着 b 点是否也会形成一个簇组粗聚类过程。而其他的 c 点以及 d 点不是簇数据点，因此 c 点以及 d 点为彼此独立的核心数据点，或形成非核心数据点。同时，在对核心数据点进行簇组并行计算时，因 c 点和其余的核心数据点不满足上面的距离规则，在聚类分析时不必对这些点进行合并计算。



(a)二维数据散点图 (b)最终结果

图 5 二维数据散点图及最终结果示意图

Fig. 5 Diagram of two-dimensional data scatter plots and final result

M 函数在上述聚类计算中主要起粗聚类的作用，不仅能将核心数据点以及该核心点的领域结合形成一组粗聚类数据组，而且能将非核心的数据点构造成独立的簇组粗聚类数据组。M 函数的程序代码如下：

```
Map:
Input:data {q^1,…,q^λ}
if(exits unvisited point q^n)
visits q^n;
if(q^n is core point)
  q^n and neighbours (q^n,Eps) are a cluster;
  q^n is core center point;
end
end
for (n=1 to λ)
if(q^n is not in a cluster)
  q^n can be a cluster;
  q^n is non core center point;
end
end
output:〈ID,cluster result〉
```

基于 DBSCAN 聚类算法进行并行计算，对各类数据进行识别计算的过程如图 6 所示。其主要的计算步骤如下：

(1)将并行计算得到的数据进行特征提取，得到 λ 组的特征参数(TZ_1，TZ_2)；

(2)基于 DBSCAN 聚类算法对数据组边界进行划分，得到具有边界的数据集合；

(3)得到一组高密度簇的核心数据组；

(4)计算得到(TZ_1，TZ_2)理论值的距离，并利用距离值对其进行判断；

(5)完成对全局数据组的判断辨识过程。

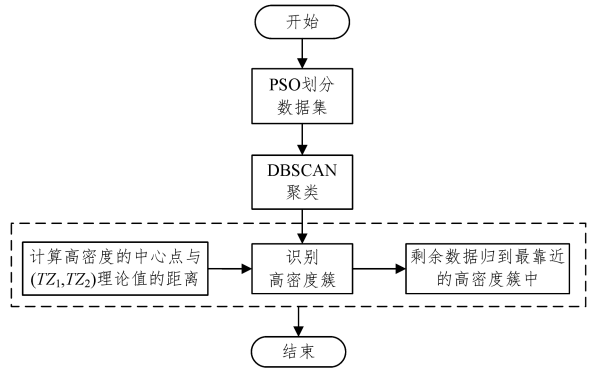


图 6 DBSCAN 算法识别流程图

Fig. 6 Identification flow chart of DBSCAN algorithm

5 结果分析

5.1 实验环境

本次实验环境主要为 3 台英特尔酷睿 4 代 CPU、2.5GHz、8GB 内存的计算机(CentOS6.5 操作系统)，选择其中一台计算机为管理层(master)节点，这台计算机也能够同时当作工作(worker)节点，两者基于一个共同的交换机构建起一个简单的局域网，选用的软件平台为 Hadoop2.4.0 以及 Spark 1.1.0 版本。

本次实验的原始数据通过 API 接口，直接从因特网上进行数据的抓取。这些数据主要是在国际时事、体育新闻、社会重大事件和娱乐新闻等 10 个板块中抓取的实验数据。实验数据源于 7880 篇新闻报道。在进行 DBSCAN 聚类计算前，采用由中科院自主开发的文本分词软件对采集到的各类数据进行了分类处理，文本中采集的数据可以基于矩阵向量的形式进行存储以及演示。

5.2 实验结果

DBSCAN 聚类算法在 Spark 平台上对数据进行聚类分析处理时，在实验过程中，DBSCAN 聚类算法对选取的 eps 以及 MinPts 的值域 jinx 进行优化分析，以此得到 DBSCAN 聚类算法的最佳数据聚类处理计算方法，最终得到单个数据点与多个数据核心点的聚类处理速度(加速比)，并以此评价 DBSCAN 聚类算法在聚类处理方面的优劣。

基于 DBSCAN 聚类算法对上述的 7880 篇新闻数据进行聚类分析，结果如图 7 所示。图 6 表明了 DBSCAN 聚类算法在 Spark 平台上对数据回归时的准确率较高，但是仍存在一些新闻数据采集不到的现象，这主要是由于新闻数据量比较大，导致 DBSCAN 聚类算法对其进行聚类簇的数量急剧增加，从图 7 中还可以发现 DBSCAN 聚类算法对娱乐性新闻数据的把握不是很好，这是因为在设定数据聚类分析时，对此类新闻数据的关键词的设定不够全面，不能对娱乐数据进行全局的数据抓取，在众多类别的新闻数据的抓取中，DBSCAN 聚类算法对体育类新闻数据的提取准确度是最高的，这是由于体育类关键词设定得较好，而娱乐性的新闻时常会出现新的关键词，应进行实时更新。尽管如此，本文算法对娱乐性新闻数据的抓取准确度也能达到 0.75 左右，进一步证明了 DBSCAN 聚类算法的稳定可靠性。

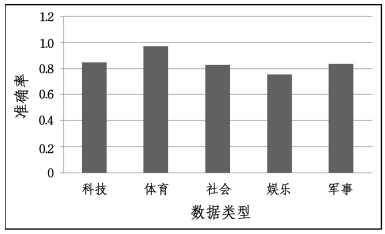


图 7 聚类结果

Fig. 7 Clustering results

为了对比分析 Spark 在单节点和多节点中的运行效率,将 Spark 分别置于单节点及多节点环境下进行执行效率的对比分析,如图 8 所示。由图 8 可知,Spark 环境下的运行时间明显短于单机的时间消耗。随着数据量的增加,多节点的运行时间并没有呈现出快速增长的趋势;而随着数据量的增大,单节点出现了运行时间激增的情况,这表明并行化处理效率更高、性能更好、实用性更强。

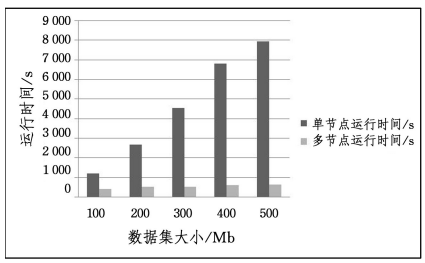


图 8 单节点和多节点环境下的算法执行效率对比

Fig. 8 Comparison of algorithm implementation efficiency in single-node and multi-node environments

图 9 给出了两种架构聚类算法的加速比比较结果。由图 9 可知,在对数据进行聚类分析时,基于 Spark 平台的 DBSCAN 聚类算法较 Hadoop 平台的加速比要大得多,DBSCAN 聚类算法在 Spark 平台上进行聚类分析时的加速比随着数据组节点数量的增加而快速增加,而在 Hadoop 平台上的加速比的增加幅度比较平缓,显示了 DBSCAN 聚类算法在 Spark 平台上对数据进行聚类分析具有较为明显的优势。

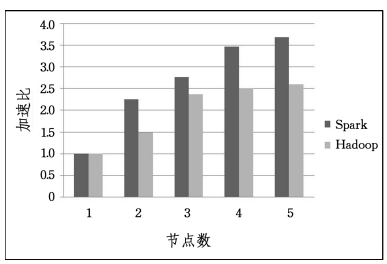


图 9 两种架构加速比比较

Fig. 9 Comparison of acceleration ratio of two architectures

结束语 本文将 DBSCAN 聚类算法与 Spark 平台相结合,基于理论仿真及实验研究对数据聚类进行分析,将改进后的 DBSCAN 聚类算法较好地应用于 Spark 平台上,并进行数据聚类的并行计算,得到如下结论。

(1)改进后的 DBSCAN 聚类算法能够大幅度降低聚类分析过程中对内存的占用率,并且能够大幅提高并行聚类分析的计算效率;

(2)基于实验验证了本文改进的 DBSCAN 聚类算法在 Spark 平台上进行聚类分析时优于 Hadoop 平台,能够获得较好的计算性能,并且比在 Hadoop 平台上的计算加速度提高了约 24.1%。

参 考 文 献

[1] AFSAR M,TAYARANI-N M H,AZIZ M. An adaptive competition-based clustering approach for wireless sensor networks [J]. Telecommunication Systems,2016,61(1):181-204.

[2] ZHAO W,XIA G S,GOU Z J,et al. An Improved DBSCAN Algorithm[J]. Journal of Sichuan Normal University (Natural Science Edition),2013(2):114-116.

[3] WANG L,WU L L,FU D M. A Density-Dased Fuzzy Adaptive Clustering Algorithm[J]. Journal of University of Science and Technology Beijing,2014(11):312-316.

[4] ZHANG Q,WANG X,WANG X. An OPTICS Clustering-Based Anomalous Data Filtering Algorithm for Condition Monitoring of Power Equipment[C]// Revised Selected Papers of the Third Ecml Pkdd Workshop on Data Analytics for Renewable Energy Integration. Springer-Verlag New York,Inc.,2015:123-134.

[5] HUANG H,YOO S,YU D,et al. Density-Aware Clustering Based on Aggregated Heat Kernel and Its Transformation[J]. Acm Transactions on Knowledge Discovery from Data,2015,9(4):29-32.

[6] YE X,SAKURAI T. Spectral clustering using robust similarity measure based on closeness of shared Nearest Neighbors[C]// International Joint Conference on Neural Networks. IEEE,2015:1-8.

[7] SINGH G,KAUR J,MULGE Y. Performance evaluation of enhanced hierarchical and partitioning based clustering algorithm (EPBCA) in data mining[C]// International Conference on Applied and Theoretical Computing and Communication Technology. IEEE,2015:805-810.

[8] WANG B,ZHANG C,SONG L,et al. Design and optimization of DBSCAN Algorithm based on CUDA[J]. Computer Science,2015,40(5):553-556.

[9] LENG Y,CHEN Z,ZHONG F,et al. BRDPHHC:A Balance RDF Data Partitioning Algorithm Based on Hybrid Hierarchical Clustering[C]// IEEE,International Conference on High PERFORMANCE Computing and Communications. IEEE,2015:1755-1760.

[10] LIN W H,TAN X J,LIU F J,et al. A new directional query method for polygon dataset in spatial database[J]. Earth Science Informatics,2015,8(4):775-786.

[11] EZUGWU A E,FRINCU M E,JUNAIDU S B. A Multiagent-Based Approach to Scheduling of Multi-component Applications in Distributed Systems[J]. Advances in Intelligent Systems and Computing,2015,347:1-12.

[12] ZHANG J,YOU S,GRUENWALD L. Large-scale spatial data processing on GPUs and GPU-accelerated clusters[J]. Sigspatial Special,2015,6(3):27-34.

[13] YANG J. From Google File System to Omega:A Decade of Advancement in Big Data Management at Google[C]// IEEE First International Conference on Big Data Computing Service and Applications. IEEE,2015:249-255.

(下转第 443 页)