CN 43-1258/TP
ISSN 1007-130X

计算机工程与科学
Computer Engineering & Science

第 39 卷第 1 期 2017 年 1 月
Vol. 39, No. 1, Jan. 2017

文章编号:1007-130X(2017)01-0035-07

基于 Spark 的 BIRCH 算法并行化的设计与实现*

李　帅[1],吴　斌[2],杜修明[3],陈玉峰[3]

(1.北京邮电大学智能通信软件与多媒体北京重点实验室,北京 100876);
2.北京邮电大学计算机学院,北京 100876;3.国网山东省电力公司电力科学研究院,山东 济南 250000)

摘　要:在分布式计算和内存为王的时代,Spark 作为基于内存计算的分布式框架技术得到了前所未有的关注与应用。着重研究 BIRCH 算法在 Spark 上并行化的设计和实现,经过理论性能分析得到并行化过程中时间消耗较多的 Spark 转化操作,同时根据并行化 BIRCH 算法的有向无环图 DAG,减少 shuffle 和磁盘读写频率,以期达到性能优化。最后,将并行化后的 BIRCH 算法分别与单机的 BIRCH 算法和 MLlib 中的 *K*-Means 聚类算法做了性能对比实验。实验结果表明,通过 Spark 对 BIRCH 算法并行化,其聚类质量没有明显的损失,并且获得了比较理想的运行时间和加速比。

关键词:Spark;BIRCH 并行化;性能优化

中图分类号:TP393.027　　**文献标志码**:A

doi:10.3969/j.issn.1007-130X.2017.01.004

Design and implementation of BIRCH algorithm parallelization based on Spark

LI Shuai[1], WU Bin[2], DU Xiu-ming[3], CHEN Yu-feng[3]

(1. Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia, Beijing University of Posts and Telecommunicaions, Beijing 100876;
2. School of Computer Science, Beijing University of Posts and Telecommunications, Beijing 100876;
3. State Gride Shandong Electric Power Research Institute, Jinan 250000, China)

Abstract: In the era when distributed computing and memory highly count, the technology of memory-based distributed computing framework, such as Spark, has gained unprecedented attention and is widely applied. We design and implement the BIRCH algorithm parallelization based on Spark, which can maximize performance optimization and reduce the frequency of shuffling and disk accessing. We do some theory analysis and describe the DAG of the BIRCH based on Spark. Finally, we compare the performance of the parallelized BIRCH algorithm with the BIRCH algorithm of a single machine and the MLlib *K*-Means clustering algorithm. Experimental results show that the parallel BIRCH algorithm based on Spark obtains ideal running time and speedup without obvious clustering quality loss.

Key words: Spark; BIRCH parallelization; performance optimization

1　引言

聚类技术被用作描述数据,衡量不同数据源间的相似性,以及把数据源分类到不同的簇中,其在数据分析、市场营销、企业决策等方面也起着越来越重要的作用。在商务上,聚类能帮助市场分析人员从客户基本库中发现不同的客户群,并且用购买模式来刻画不同的客户群的特征。在生物学上,聚类能用于推导植物和动物的分类,对基因进行分

* **收稿日期**:2016-09-05;**修回日期**:2016-11-12
基金项目:国家 863 计划(2015AA050204);国网科技项目(60873120)
通信地址:100876 北京市海淀区北京邮电大学计算机学院
Address: School of Computer Science, Beijing University of Posts and Telecommunications, Haidian District, Beijing 100876, P. R. China

类,获得对种群中固有结构的认识。聚类在地球观测数据库中相似地区的确定,汽车保险单持有者的分组,及根据房子的类型、价值和地理位置对一个城市中房屋的分组上也可以发挥作用。聚类也能用于对 Web 上的文档进行分类,以发现信息。现有的聚类算法一般分为以 K-Means[1] 为代表基于划分的聚类、以 DBSCAN(Density-Based Spatial Clustering of Applications with Noise)[2] 为代表基于密度的聚类、以分层的平衡迭代归约及聚类 BIRCH(Balanced Iterative Reducing and Clustering using Hierarchies)[3] 为代表基于层次的聚类和以 STING(STatistical INformation Grid)[4] 为代表的基于网格的聚类。

同时,随着数据量的不断增大和时代的发展,单台计算机的计算能力已无法满足社会生产生活的需要,类似 Hadoop、Spark 等分布式大数据处理框架技术应运而生。现今的许多企业、政府、学校和其他的一些社会组织都应用了大数据处理框架技术来处理大数据,用于挖掘数据中的价值,分布式计算已经成为各个领域不可缺少的技术之一。

为更好地统计分析海量数据,聚类算法的并行化成了必然的趋势。Tsapanos 等人[5] 提出了基于 MapReduce 和核函数的 *K*-Means, Patwary 等人[6] 提出了并行的 DBSCAN, Chierichetti 等人[7] 做了基于 MapReduce 的关联聚类以及 Jin 等人[8] 提出的增量式的基于 MapReduce 的单联层次聚类等,这些聚类算法都很好地对原算法做出了改进和并行化。层次聚类算法 BIRCH 虽然可以在随意给定的内存中运行,但是随着数据量的增大,聚类精度也会随之降低,加之 BIRCH 算法需要建立类似 B 树的数据结构,数据需要串行输入,本身不易并行化,所以尽管 Hadoop 和 Spark 等技术已经相当成熟,但是并没有基于这些技术对 BIRCH 算法的并行化。然而,就 BIRCH 算法而言,它具有很多其他聚类算法不可比拟的优点,良好的数据压缩性,能够极大地节约内存和 I/O 的资源,能够发现离群点和有效处理高维数据等。所以,如果能结合现有的基于内存的分布式大数据处理框架技术 Spark 对 BIRCH 算法进行并行化,意义重大。同时,虽然 Spark 中自带了机器学习库 MLlib[9],使一般应用领域的数据分析人员也可以方便地使用 Spark 的并行化编程接口和强大的计算能力[10],但是 MLlib 库中并没有 BIRCH 算法,本文所做的工作也将是对 Spark 机器学习库的一个扩充。

本文的主要工作是通过 Spark 对 BIRCH 算法进行并行化。其思路主要是利用 BIRCH 算法压缩数据集和 *K*-Means 算法可以并行化[11] 的特点,将两者相结合使用[12]。同时,本文还对并行化后的 BIRCH 算法进行了理论分析,并找出了主要的时间开销,且设计了 Spark 执行并行 BIRCH 算法时的有向无环图 DAG(Directed Acycline Praph),做了若干并行 BIRCH 算法的优化工作。最后,本文将并行 BIRCH 与单机 BIRCH 以及 Spark 机器学习库中自带的 *K*-Means 算法做了性能对比实验,并对并行化算法的性能进行了评定。

2 相关工作介绍

聚类算法 BIRCH 来源于 Zhang、Ramakrishnan 和 Livny 的工作[3],他们创建的 BIRCH 算法的最终结果是建立一棵类似 B 树的聚类特征树。聚类特征 CF(Cluster Feature)是 BIRCH 算法的核心概念。CF 正是采用三元组(N,LS,SS)的方式才达到了压缩数据集的效果,从而使 BIRCH 算法能够在有限的内存和低 IO 开销的情况下得以运行。也正是源于 BIRCH 算法的这些优势,越来越多的研究者也将目光投向了 BIRCH 算法的并行化研究。余晓山[13] 提出了基于数据划分的 BIRCH 聚类,朱映辉、江玉珍[14] 提出了通过自定义数据类型的方式进行 BIRCH 算法的并行化,以及韦相[15] 提出的基于密度的改进 BIRCH 聚类算等。其中,Garg 等人[12] 在 2006 年研究提出的 PBIRCH 算法是基于 MPI 集群环境对 BIRCH 算法的并行化,本文的并行路线借鉴了该篇文献。

Apache Spark 是一个围绕速度、易用性和复杂分析构建的大数据处理框架[16]。Spark 提供了一个全面、统一的框架用于管理各种有着不同性质(文本数据、图表数据等)的数据集和数据源(批量数据或实时的流数据)的大数据处理的需求,利用内存数据存储和接近实时的处理能力,Spark 比 Hadoop 要快很多倍[17]。本文将充分利用 Spark 的优势来对 BIRCH 并行化,以期达到性能最优。

3 Spark 对 BIRCH 的并行化算法

本文将 Spark 并行化过的 BIRCH 算法称为 SparkBIRCH。下面将分别对 SparkBIRCH 的执行过程、时间开销和优化一一描述和分析。

3.1 SparkBIRCH 算法并行化的策略

如图 1 所示,SparkBIRCH 算法主要分为两

步:

(1)分区内建立聚类特征树(CF 树),并以 CF 树的叶子结点作为新的数据点,最后随机选择 K 个点作为 K-Means 初始的类中心点,在这一步中,数据量得到了极大的压缩;

(2)对产生的新数据点应用 K-Means 聚类算法,使 BIRCH 算法的并行化得以实现。

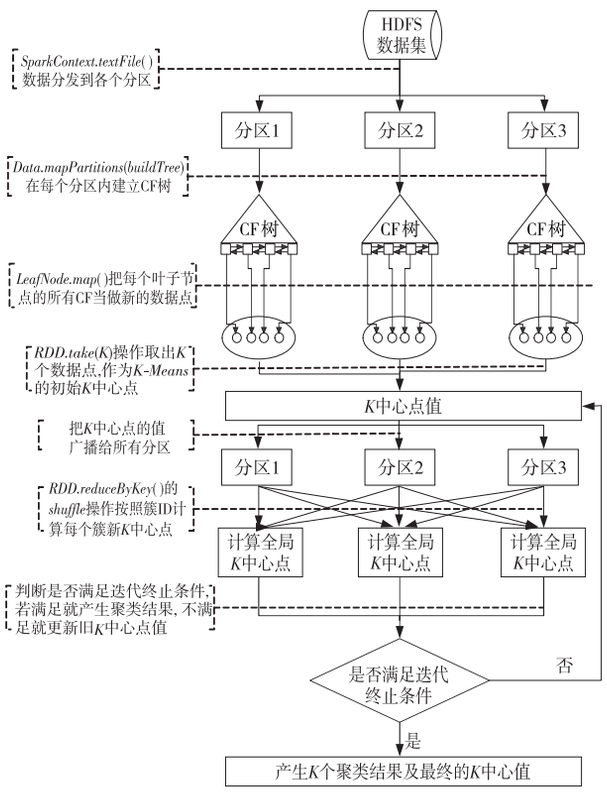

Figure 1　Logic execution graph of the SparkBIRCH

图 1　SparkBIRCH 算法的逻辑执行示意图

首先,Spark 通过读取 Hadoop 分布式文件系统 HDFS(Hadoop Distributed File System)上的数据并分发数据到每个分区,然后在每个分区内建立一棵属于该分区的聚类特征树。在分区内建树的过程与单机的情况是一样的。

建好树之后,每个分区的聚类特征树的叶子结点上都会有 BIRCH 算法指定数目的聚类特征 CF。为了可以将 BIRCH 算法并行化,同时考虑到 K-Means 良好的并行性,本文将这些新生成的叶子结点的 CF 作为新的数据点应用于 K-Means 算法。可以想到,这些 CF 的数目是远远小于原始数据点数的,数据集得到了很大程度的压缩。

应用 K-Means 算法时,首先随机取出 K 个新数据点,然后广播到每个分区中。每个分区以此 K 个点作为 K-Means 算法的初始类中心点,并在分区内部对每一个数据点划分所属簇,之后重新计算 K 个类中心点,再由主结点广播新的 K 个类中心点到所有结点。这仅是一次 K-Means 聚类操作。之后,迭代聚类,每次迭代重新计算 K 个类中心点的值,直到上一次的 K 个类中心点值与本次 K 个类中心点值之差小于指定的阈值,迭代终止。

3.2　性能分析

设原数据集行数为 m_{raw},BIRCH 数据集压缩比为 a,新数据集行数为 m_{new},每行的数据大小为 C,集群结点数为 S,第 i 台机器为 S_i,集群各机器之间的通信带宽为 B,总的 shuffle 次数为 f,第 i 台机器处理数据的总时间为 $T(S_i)$,其它的一些时间花销为 $\varepsilon(\text{other})$。其中,

$$m_{\text{new}} = m_{\text{raw}} * a \tag{1}$$

其中,a 的大小与 BIRCH 算法的参数设置以及数据的分布有关系。

同时假设每台机器的数据都会 shuffle 到其它结点,并且数据量不变,所以总的花销时间 $TotalTime$ 为:

$$TotalTime = \frac{(S-1) * m_{\text{new}} * C * f}{B} + \max\{T(S_i), 1 \leqslant i \leqslant S\} + \varepsilon(\text{other}) = Shuffle(S) + Compute(S) + OtherTimeCost \tag{2}$$

其中,$Shuffle(S)$ 为 shuffle 过程的总时间,$Compute(S)$ 为计算的最大开销。所以 $TotalTime$ 为实际时间开销 $ActualTime$ 的上限。即:

$$ActualTime \leqslant TotalTime \tag{3}$$

由公式(2)可以知道,在数据量一定的情况下,shuffle 的时间是与集群结点数 S 成正比的,每台机器的计算时间与集群结点数 S 是成反比的。

如图 2 所示,本文要做的主要工作之一就是减少 shuffle 的花费时间,尽量寻求最优的 shuffle 时间开销。

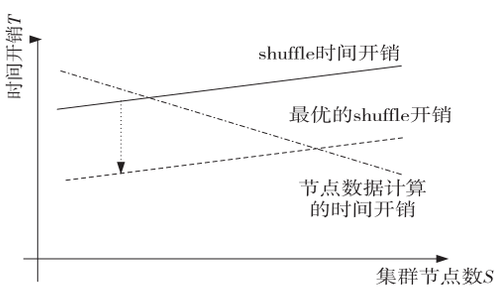

Figure 2　Time expenditure of shuffle and data calculating

图 2　shuffle 和数据计算时间开销示意图

3.3　SparkBIRCH 算法的优化

由上节理论上的性能分析可以看出,如果 Spark 并行化的程序有太多的 shuffle 或通信的操

作就会造成严重的性能损失,所以在进行并行算法的设计时就必须尽量减少 shuffle 和通信的次数。SparkBIRCH 算法的物理 DAG 图如图 3 所示。

SparkBIRCH 算法的应用程序至少产生了两个作业 job,如图 3 所示,实线贯穿的过程是一个 job,称为 *job*_0,虚线贯穿的过程是另一个 job,称为 *job_i*。*job*_0 是 *take*()函数触发的,*job_i* 是 *K*-Means 部分的 *collectAsMap*()函数产生的,其中 $i\geqslant 1$,i 的值取决于迭代次数,也就是取决于迭代终止条件。

*job*_0 只包含一个阶段,没有 shuffle 操作。这个 job 主要的工作是在每个分区内建立一棵 CF 树,并把所有分区的 CF 树的叶子结点中的 CF 作为新的数据点,最后用 *take*() 函数取出 *K* 个数据作为 *K*-Means 算法的初始类中心点。在这个 job 中没有产生 shuffle 过程,且每一次转化操作都会被下一步用到,此 job 得到了优化。

第 i 个作业 job 包含两个阶段,有一次 shuffle 操作,一次是 *K* 个聚类中心点的广播操作。这个 job 主要的工作是执行以新数据点作为输入的 *K*-Means 算法,并更新 *K* 中心点的值。正如虚线标注的那样,这些步骤要执行多次。另外,因为深色圆圈标示的转化操作是不会被改变且每次迭代都会用到的,所以可以 Cache 到内存,以最大化减少磁盘读写次数。*K*-Means 算法中的每一轮迭代肯定会产生至少一个 shuffle,而在这个 job 中,每一轮迭代只有一个 shuffle 产生,此 job 得到了优化。

本文按照 Spark 有向无环图 DAG 的形式再次优化 SparkBIRCH 程序,可以最大程度地降低 shuffle 和通信操作的频率,以及最大程度地减少读写磁盘的次数,加快算法的执行速度。

4 实验

本文以 HDFS 为存储系统的平台,应用真实数据集,测试 Spark 集群环境下的并行 BIRCH 算法的性能,并与单机 BIRCH 算法、Spark MLlib 自带的 *K*-Means 算法做了对比实验。

4.1 实验环境

Spark 集群包括 3 台 slave 结点,1 台 master 结点。服务器是 Dell R730,CPU 信息是 Intel(R) Xeon(R) CPU E5-2609 v3,每台机器配有 2 个处理器,以太网卡是 Broadcom Corporation NetXtreme BCM5720 Gigabit Ethernet PCIE * 4,硬盘读写速度 375.29 MB/s,16 GB 运行内存。操作系统为 64 位 centos 6.5,JDK 版本是 jdk 1.7,Hadoop 版本是 2.6.4,Spark 版本是 1.5.1。

测试 SparkBIRCH 算法准确度时,数据集采用东芬兰大学提供的 R15 数据集[18]、D31 数据集[18]和 Aggregation 数据集[19]。R15 数据集有 15 个簇,每个簇有 40 个点,共包含 600 * 2(600 行,2 列)数据,每行数据都带有簇类标签;D31 数据集有 31 个簇,每个簇有 100 个点,共包含 3100 * 2 数据,每行数据都带有簇类标签;Aggregation 数据集有 7 个簇,每个簇的点数非均匀分布,共包含 787 * 2 数据,每行数据都带有簇类标签。

测试 SparkBIRCH 的执行速度时,数据集采用人工合成数据,生成方式为随机生成,数值范围

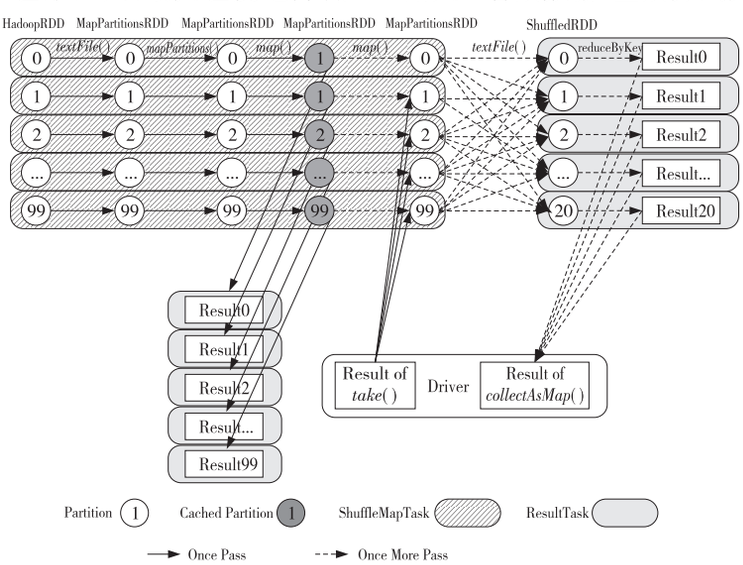

Figure 3 Physical DAG of the SparkBIRCH

图 3 SparkBIRCH 算法的物理 DAG 图

为(0,10 000)。本文分别采用 100 万行、1 000 万行、2 000 万行、4 000 万行的数据集,并与单机 BIRCH 算法、Spark MLlib 自带的 K-Means 算法作了对比。

4.2　准确度实验

本文就 R15、D31、Aggregation 数据集分别做了 SparkBIRCH 算法和单机 BIRCH 算法的准确度对比实验,并得到了表 1 所示的结果。

Table 1　Clustering accuracy of SparkBIRCH and BIRCH

表 1　SparkBIRCH 算法与单机 BIRCH 算法聚类准确度

	SparkBIRCH/%	单机 BIRCH/%
R15	99.33	99.66
D31	93.03	95.16
Aggregation	91.11	94.83

为了可以更形象地对 SparkBIRCH 进行准确度分析,本文图形化了原始数据集和聚类后数据集。图 4～图 6 分别是 R15、D31、Aggregation 的原始数据集分布效果图。图 7～图 9 是对应的聚类结果分布效果图,图中圆圈的内容是没有得到正确聚类的数据点或错误点比较集中的地方。通过将三种数据集的原始数据集和聚类效果图作比较可以看出,聚类错误的点处于两个簇边缘,说明 SparkBIRCH 算法具有很高的可信度。

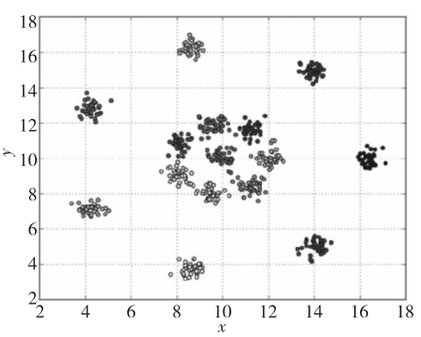

Figure 4　Distribution of R15

图 4　R15 数据集分布

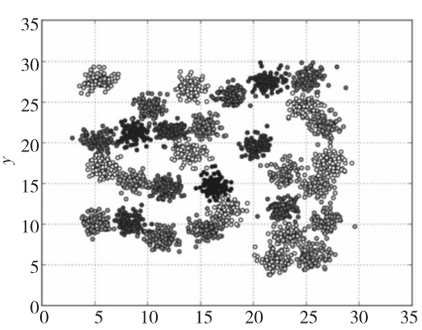

Figure 5　Distribution of D31

图 5　D31 数据集分布

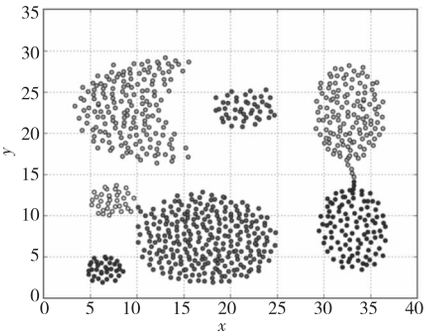

Figure 6　Distribution of Aggregation

图 6　Aggregation 数据集分布

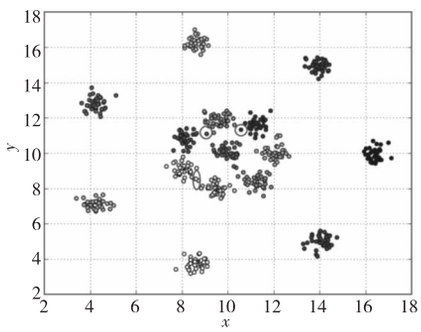

Figure 7　Clustering distribution of R15

图 7　R15 聚类结果分布

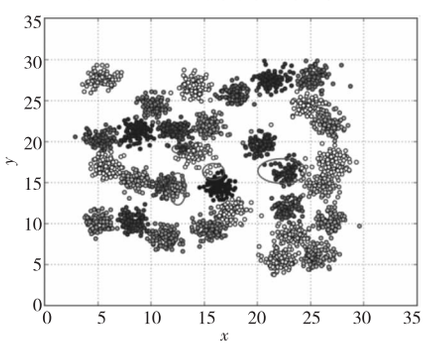

Figure 8　Clustering distribution of D31

图 8　D31 聚类结果分布

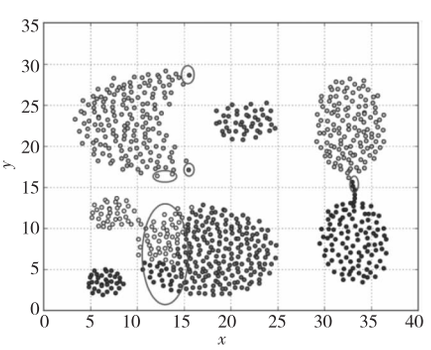

Figure 9　Clustering distribution of Aggregation

图 9　Aggregation 数据集分布

由表 1 可以看到,SparkBIRCH 的聚类精度略低于单机的 BIRCH 算法,且应用 R15 数据集时准确度最高,D31 次之,Aggregation 居最后。三

个数据集的分布图如图4～图6所示,可以看出R15和D31的每个簇都是等量的,但是R15相比D31簇之间的间隔比较远且聚类个数较少,聚类时更不容易损失准确度;同时可以明显地看出,Aggregation数据集的每个簇并不是等量的,聚类时造成的影响如图9所示,小簇可能会侵占大簇的数据点,从而使聚类准确度降低。BIRCH对非球形聚类效果不佳[15],如图9所示,SparkBIRCH继承了这个缺点,并没有因为并行化而改善这个状况。

4.3 性能对比实验

为了更好地进行性能对比,本文还选择了Spark MLlib中典型聚类算法*K*-Means作为性能参照,以下称之为MLlib *K*-Means。图10展示了SparkBIRCH、单机BIRCH、MLlib *K*-Means算法随着数据量增大的变化情况。可以看到,SparkBIRCH执行时间随数据量增大稍有增加,涨幅最不明显。MLlib *K*-Means算法的执行时间随数据量的增大大致呈线性增加,且斜率高于SparkBIRCH,可以预判随着数据量继续增大,MLlib *K*-Means算法的运行时间将远远高于SparkBIRCH的运行时间。单机BIRCH算法的执行时间随数据量的增大而增加的幅度是最明显的,数据量的继续增加将使算法的执行过程变得非常缓慢。

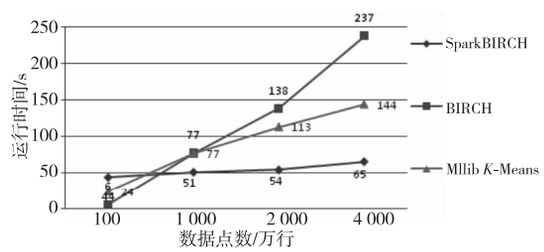

Figure 10 Running time comparison on different-sized datasets

图10 数据集不同大小下3种算法的运行时间对比

通过SparkBIRCH与单机BIRCH、MLlib *K*-Means算法对比,可以看出SparkBIRCH具有良好的速度性能。在某个区间内的数据集的增大,SparkBIRCH的执行时间不会随着出现明显大幅度的增加。同时,在数据量小的情况下,SparkBIRCH和MLlib *K*-Means的很多时间都消耗在shuffle过程中,反而不如单机情况下的执行效率,这也告诉我们小数据量并不适合在集群下运行。

4.4 加速比实验

表2展示了大数据集的加速比与CPU核数是成正比关系的,随着核数增加,加速比也随之增加。并且在核数一定的情况下,随着数据点增多,其加速比也随之增大。但是,当数据量小的时候,加速比的值不是很大,随核数增大的趋势也不明显。这主要是因为小数据量计算的时间开销比较小,而用于通信的时间开销比较大。

Table 2 Speed-up ratio of the SparkBIRCH under different numbers of cores and data lines

表2 SparkBIRCH在不同核数和不同数据行数下的加速比

核数	2	4	6
100万行	0.133	0.136	0.136
1 000万行	1.375	1.510	1.510
2 000万行	2.061	2.421	2.556
4 000万行	2.755	3.485	3.646

5 结束语

本文主要实现了BIRCH在Spark上的并行化,扩充了Spark MLlib,并理论分析了并行化过程中执行时间的主要开销,且通过设计描绘了SparkBIRCH的DAG图,找出可Cache到内存的转化操作,将执行过程读写磁盘的中间操作降到了最少。同时,根据DAG图,尽量减少算法和程序中shuffle和通信次数,重新优化了SparkBIRCH的逻辑过程,最大程度地缩短了算法执行的时间。另外,本文实验对比了SparkBIRCH与单机BIRCH的聚类质量,SparkBIRCH与单机BIRCH、MLlib *K*-Means算法的执行时间,以及SparkBIRCH随CPU核数与数据集点数的加速比。最后,实验结果表明,通过运用大数据处理框架Spark对BIRCH算法并行化,以及对其做的性能优化处理,使得聚类质量在没有明显损失的情况下,基于Spark的并行化BIRCH算法取得了比较理想的运行时间和加速比。

在聚类准确度的实验中可以看出,SparkBIRCH对于存在大小不同的簇的数据集和非球形簇的数据集聚类效果欠佳,所以今后的工作主要有两方面:一是改善SparkBIRCH算法对不同大小簇的聚类质量;二是改善SparkBIRCH算法,以提高对非球形簇的识别能力。

参考文献:

[1] Jain A K,Murty M N,Flynn P J. Data clustering: A review [J]. ACM Computing Surveys (CSUR),1999,31(3):264-323.

[2] Ester M,Kriegel H P,Sander J,et al. A density-based algorithm for discovering clusters in large spatial databases with noise[C]// Proc of Kdd,1996:226-231.

[3] Zhang T, Ramakrishnan R, Livny M. BIRCH: An efficient data clustering method for very large databases[C]// Proc of ACM Sigmod Record,1996: 103-114.

[4] Wang W, Yang J, Muntz R. STING: A statistical information grid approach to spatial data mining[C]// Proc of VLDB, 1997: 186-195.

[5] Tsapanos N, Tefas A, Nikolaidis N, et al. Efficient MapReduce kernel *k*-means for big data clustering[C]// Proc of the 9th ACM Hellenic Conference on Artificial Intelligence, 2016: 28.

[6] Patwary M M A, Palsetia D, Agrawal A, et al. A new scalable parallel DBSCAN algorithm using the disjoint-set data structure[C]// Proc of 2012 IEEE International Conference on High Performance Computing, Networking, Storage and Analysis (SC 2012), 2012: 1-11.

[7] Chierichetti F, Dalvi N, Kumar R. Correlation clustering in MapReduce[C]// Proc of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014: 641-650.

[8] Jin C, Chen Z, Hendrix W, et al. Incremental, distributed single-linkage hierarchical clustering algorithm using MapReduce[C]// Proc of the Symposium on High Performance Computing, 2015: 83-92.

[9] MLlib | Spark [EB/OL]. [2016-05-]. http://spark.apache.org/mllib/.

[10] Liu Zhi-qiang, Gu Rong, Yuan Chun-feng, et al. Parallelization of classification algorithms based on SparkR[J]. Journal of Frontiers of Computer Science and Technology, 2015, 9(11): 1281-1294. (in Chinese)

[11] Meng X, Bradley J, Yavuz B, et al. MLlib: Machine learning in apache spark[J]. The Journal of Machine Learning Research, 2016, 17(1): 1235-1241.

[12] Garg A, Mangla A, Gupta N, et al. PBIRCH: A scalable parallel clustering algorithm for incremental data[C]// Proc of 2006 10th International Database Engineering and Applications Symposium (IDEAS'06), 2006: 315-316.

[13] Yu Xiao-shan, Wu Yang-yang. Parallel text hierarchical clustering based on MapReduce[J]. Journal of Computer Applications, 2014, 34(6): 1595-1599. (in Chinese)

[14] Zhu Ying-hui, Jiang Yu-zhen. Research of BIRCH clustering algorithm optimization and parallelism[J]. Computer Engineering and Design, 2007, 28(18): 4345-4346. (in Chinese)

[15] Wei Xiang. Improved BIRCH clustering algorithm based on density[J]. Computer Engineering and Applications, 2013, 49(10): 201-205. (in Chinese)

[16] Holden K, Andy K, Patrick W, et al. Learning spark: Lighting-fast data analysis[M]. New York: O'Reilly Media, 2015.

[17] Zaharia M, Chowdhury M, Franklin M J, et al. Spark: Cluster computing with working sets[C]// Proc of HotCloud'10, 2010: 10.

[18] Veenman C J, Reinders M J T, Backer E. A maximum variance cluster algorithm[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002, 24(9): 1273-1280.

[19] Gionis A, Mannila H, Tsaparas P. Clustering aggregation [J]. ACM Transactions on Knowledge Discovery from Data (TKDD), 2007, 1(1): 1-30.

附中文参考文献:

[10] 刘志强,顾荣,袁春风,等. 基于 SparkR 的分类算法并行化研究[J]. 计算机科学与探索,2015,9(11):1281-1294.

[13] 余晓山,吴扬扬. 基于 MapReduce 的文本层次聚类并行化[J]. 计算机应用,2014,34(6):1595-1599.

[14] 朱映辉,江玉珍. BIRCH 聚类算法优化及并行化研究[J]. 计算机工程与设计,2007,28(18):4345-4346.

[15] 韦相. 基于密度的改进 BIRCH 聚类算法[J]. 计算机工程与应用,2013,49(10):201-205.

作者简介:

李帅(1992 -),男,山东滨州人,硕士生,研究方向为机器学习和分布式计算。**E-mail**:1105785283@qq.com

LI Shuai, born in 1992, MS candidate, his research interests include machine learning, and distributed computing.

吴斌(1969 -),男,湖南长沙人,博士,教授,CCF 会员(E200010415S),研究方向为数据挖掘、复杂网络、大数据与云计算、商务智能。**E-mail**:wubin@bupt.edu.cn

WU Bin, born in 1969, PhD, professor, CCF member(E200010415S), his research interests include data mining, complex network, big data and cloud computing, and business intelligence.

杜修明(1985 -),男,湖北襄阳人,硕士生,工程师,研究方向为变压器类设备状态检修。**E-mail**:xiumingd@gmail.com

DU Xiu-ming, born in 1985, MS candidate, engineer, his research interest includes condition-based maintenance of transformer.

陈玉峰(1970 -),男,山东枣庄人,高级工程师,研究方向为变压器类设备状态检修和电力大数据分析。**E-mail**:chenyufeng169@sina.com

CHEN Yu-feng, born in 1970, senior engineer, his research interests include condition-based maintenance of transformer, and big data analytics of power systems.