DOI: 10.13718/j.cnki.xdzk.2016.11.028

K-Means 算法的 MapReduce 并行实现①

蒋　溢，　刘鑫洋

重庆邮电大学 计算机学院，重庆 400065

摘要：首先搭建了 Hadoop 集群，然后在 MapReduce 模型中实现 K-Means 算法，最后设计多组实验验证了算法的稳定性和准确率.

关　键　词：Hadoop 分布式计算；K-Means；MapReduce

中图分类号：TP311　　**文献标志码**：A　　**文章编号**：1673-9868(2016)11-0180-06

Hadoop(http://hadoop.apache.org)属于开源的分布式文件系统集群，根据谷歌的 GFS 的思想来进行实现. HDFS 具有高容错、高吞吐量数据访问以及适合大规模部署等特点. 用户可以在不清楚底层运作的情况下，直接进行 MapReduce 并行编程，从而实现将普通程序运行在多个节点组成的 Hadoop 集群上，由此充分利用集群的性能，对大量数据进行处理，实现高性能地运算和存储.

MapReduce 属于并行编程的模型. MapReduce 通常被用来并行处理海量数据，把对一个大数据集的操作割成对多个小数据集的操作，最后再把各个小数据集的操作结果合并起来得到最终的结果. MapReduce 的名字来源于它的两个最为重要的操作过程：Map 和 Reduce. Map 是将一组数据(数据形式是〈key，value〉)一对一地映射为另外一组数据(数据形式也是〈key，value〉)，而 Reduce 是将 Map 的输出进行归约，最后输出结果同样是〈key，value〉的形式.

1　K-Means 算法

1.1　K-Means 算法简介

K-Means 算法能够基于用户所设定的 K 值将数据划分为 K 个簇，进而以距离为标准确定各样本的相似度，具有简单、高效的特性[1]，算法的实现步骤如下：

1) 随机地选择 K 个初始中心点；

2) 根据相似度进行分类；

3) 重新计算每个类的中心；

4) 根据相似度重新分类；

5) 相似度重新分类，到误差平方总和收敛或迭代次数达到上限.

对于第 2)步的相似度的衡量，通常采用样本点的距离来作为标准，即距离越近被认为相似度更大，距离越远，相似度越小. 对于距离的计算通常采用欧式距离(1)、曼哈顿距离(2)、闵可夫斯基距离(3)：

$$d=\sqrt{\sum(X_{1i}-X_{2i})^2} \tag{1}$$

$$d=\sum\mid X_{1i}-X_{2i}\mid \tag{2}$$

① 收稿日期：2016-03-12

基金项目：工信部物联网发展专项基金项目(2012-2-5)；重庆市教委科学技术研究项目(KJ1400414).

作者简介：蒋　溢(1969-)，男，湖北安陆人，教授级高级工程师，硕士生导师，CCF 会员 (E200016596M)，主要从事网络信息安全的研究.

$$d=\sqrt[p]{\sum(X_{1i}-X_{2i})^2} \tag{3}$$

另外，通常使用各个簇的误差平方和的总和作为聚类结果准确性的衡量标准，被称为准则函数，其计算公式如下：

$$E=\sum_{i=1}^{k}\sum_{X\in C_i}|X-\overline{X}|^2 \tag{4}$$

其中：E 表示各个簇内的误差平方和的总和，k 表示簇的总个数，C_i 表示一个簇中所有元素的集合，$\overline{X}$ 表示 C_i 的均值中心.

2 K-Means 并行化设计

一个 MapReduce 模型还需要用户编写整个 MapReduce 的运行逻辑、Map 函数以及 Reduce 函数[2]. 比如在 Map 阶段，用户可以重写 setup 函数，以便在执行 map 前完成一些必要的工作，亦可重写 cleanup 函数，在执行完 map 函数后做一些需要的工作，或者设置 MapReduce 函数迭代执行. 在本文的 K-Means 算法就需要设置 MapReduce 程序迭代执行，以不断优化初始中心点. 另外配置 MapReduce 程序的运行环境也需要用户自己进行.

在对 K-Means 实现 MapReduce 编程的过程中，把这整个过程分为两个部分：首先，采用一 MapReduce 程序，用以实现抽样时的最大最小距离法，得到首个聚类的初始中心点；其次进行 K-Means 的迭代优化初始中心点，每次迭代均为一次 MapReduce 过程，直到结果收敛便结束整个 MapReduce 过程[3].

1) 数据输入阶段. 由于将 Hadoop 框架的自动分割作为 MapReduce 过程的输入，因此取样阶段不需要人工干预，使用其默认的分割作为一个样本即可；

2) Map 阶段. 此阶段输入格式为$\langle K_1, V_1\rangle$，K_1 为数据的偏移量，V_1 为样本数据. 在此 Map 函数主要是基于上述的最大最小距离法求得 K 个初始中心点. 输出格式为$\langle K_2, V_2\rangle$，K_2 设置为 1，没有具体意义，$V2$ 为各个初始中心点的各维值，共 K 行，每行一个点；

3) Reduce 阶段. 此阶段输入为$\langle K_2, list(V_2)\rangle$. Reduce 函数将 list 中的所有 V_2 集合成一个点集，然后使用最大最小距离法得到 K 个中心点，输出格式为$\langle K_3, V_3\rangle$，K_3 没有意义，本文设置为 1，V_3 储存 K 个初始中心点，每个点占用一行.

如此，便找到了首批初始中心点. 下面进行 K-Means[4] 迭代优化中心点的过程

1) 据输入阶段. 将待处理的数据集由 Hadoop 框架分割后输入到各个 Map 上进行处理. 输出格式为$\langle K_1, V_1\rangle$，K_1 为数据偏移量，V_1 为一行数据，该行数据是由实验数据集中的一个数据项的各维值组成的一个字符串(本文采用的数据是人工构造的数据集，这一数据集又包含有多个 40 维数据项；

2) Map 阶段. 此阶段输入为上一阶段输出. Map 函数根据欧式距离最小原则对数据点进行分类，输出格式为$\langle K_2, V_2\rangle$，K_2 为点所属的簇的编号，V_2 为一个复合字符串(Hadoop 为 Text 类型，即 Java 中的 String)，包含该点距离中心的距离以及该点的各维数值；

3) Combine 阶段. 此阶段在本地对 Map 阶段输出进行预处理，以减少数据在网络中的流动量. 其输入格式为$\langle K_2, \text{list}(V_2)\rangle$，Reduce 函数主要是对本地的 K_2 值相同的点，将它们的各维值进行累加，计算所有点的误差平方和，输出格式为$\langle K_3, V_3\rangle$，K_3 为所属簇的编号，V_3 为一个复合字符串，包含了误差平方和、累加的点的个数以及各点的各维对应相加后的结果；

4) Reduce 阶段. 此阶段将上一阶段$\langle K_3, V_3\rangle$的输出经处理后形成的$\langle K_3, \text{list}(V_3)\rangle$作为输入. 经过 Reduce 函数处理后，输出格式为$\langle K_4, V_4\rangle$，K_4 为该簇的编号，V_4 为一个复合的字符串，包含该簇的误差平方和以及该簇新的中心的各维的值；

5) 数据输出阶段. 此时将上一阶段的输出存至 HDFS[5]，在后续的迭代工作中再次使用；

6) 执行完一次 MapReduce 程序后，需要根据新的中心点重新分类并计算误差平方和，与旧的误差平方和比较后，对于结果进行判断，看其收敛性，如果结果是收敛的，便不再进行迭代，否则仍旧重复迭代过程.

3　优化前后 K-Means 算法性能对比

本节就算法的稳定与准确性展开了针对性的实验工作，期间所涉及到的各类数据来自于 UCI 数据库[6]. 在此实验中，所使用的主要是被广泛应用的 Wine 与 Iris 数据集. 后者又包含有 150 个数据项，而每个数据项又有着 4 个属性，分别为花萼的长度与宽度、花瓣的长度与宽度. 根据这 4 个属性将 150 个数据项分成 3 大类，分别 IrisSetosa(山鸢尾)，Iris Versicolor(变色鸢尾)和 Iris Virginica(维吉尼亚鸢尾)，每类各 50 个数据项. 另一个数据集 Wine，总共包含有 178 个数据项，每个数据项均含有 14 个属性，包括酒精含量、羟基丁二酸、总酚、色相等. 根据这 14 个属性将这些样本分为了 3 大类，本文将其取名一类酒、二类酒、三类酒.

首先使用未经改进的 k-means 算法进行实验，经过重复 20 次实验，取其中出现频率远高于其他实验结果出现频率的一种实验结果展示见表 1.

表 1　k-means 算法在 Iris 数据集实验结果

聚　类	名　　称			总　数
	山鸢尾	变色鸢尾	维吉尼亚鸢尾	
1	50	0	0	50
2	0	40	8	48
3	0	10	42	52

由表 1 可以计算得出，改进前的 k-means 算法准确率为

$$\frac{50+40+42}{150}=0.88$$

再使用改进后的 k-means 算法进行实验，同样实验 20 次，取其中出现频率远高于其他实验结果出现频率的一次实验结果，如表 2 所示.

表 2　改进后的 k-means 算法 Iris 数据集实验结果

聚　类	名　　称			总　数
	山鸢尾	变色鸢尾	维吉尼亚鸢尾	
1	50	0	0	50
2	0	47	14	61
3	0	3	36	39

同样由表 2 计算得出改进后的 k-means 算法准确率为

$$\frac{50+47+36}{150}\approx 0.886\ 7$$

再将改进前后两种算法分别运行 20 次，计算出每次聚类的准确率. 准确率计算公式如下：

$$\text{准确率}=\frac{\text{归类正确的总数}}{\text{样本总数}}$$

准确率的计算结果汇总后绘制成(图 1).

由图 1 可以看出，改进后的 k-means 算法，准确率一直稳定地保持在 88.67%，而改进前的 k-means 算法准确率维持在 88%左右，并且准确率并不稳定. 通过对比发现，无论是准确率还是稳定性，改进之后的算法明显要更具优势. 而改进前的算法之所以没有稳定的准确率，关键在于确定初始中心点时太过于随机，若是在该环节过于相似，那么聚类也就不可能会有理想的结果. 采用上述相同的实验方式，对数据集 Wine 的实验结果见表 3、表 4 和图 2.

表 3　k-means 算法在 Wine 数据集实验结果

聚　类	名　　称			总　数
	酒 1	酒 2	酒 3	
1	59	6	0	65
2	0	62	0	62
3	0	3	48	51
总计	59	71	48	178

表4 改进后的k-means算法在Wine数据集实验结果

聚 类	名 称			总 数
	酒1	酒2	酒3	
1	59	2	0	61
2	0	65	0	65
3	0	4	48	52
总计	59	71	48	178

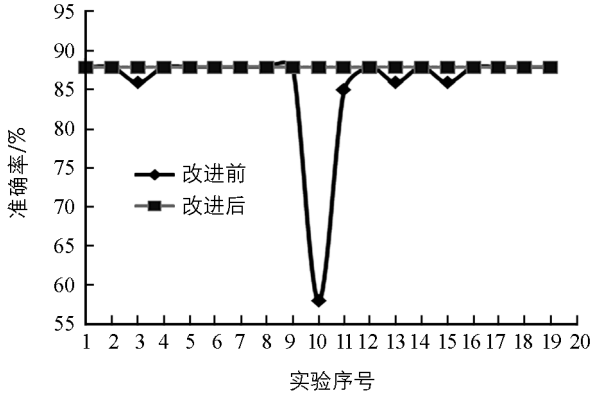

图1 改进前后的K-Means算法在Iris数据集上的实验对比图

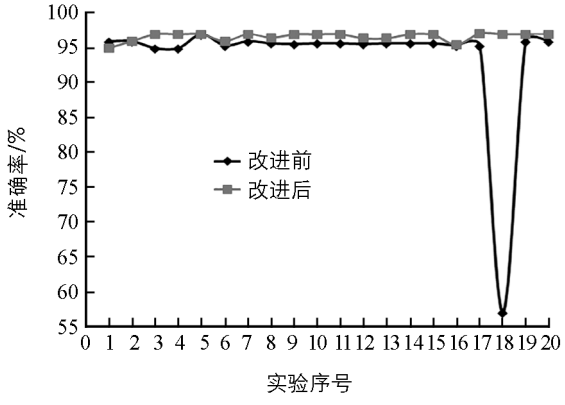

图2 改进前后的K-Means算法在Wine数据集上的实验对比图

由表3和表4可以计算得出改进前算法准确率为：

$$\frac{59+62+48}{178}=0.949\ 4$$

改进后算法准确率为：

$$\frac{59+65+48}{178}=0.966\ 3$$

计算结果表明改进后算法的准确率更高，但从整体上来看，即便是改进之前其准确率也是有着较高水平的. 通过图2可以得出结论，改进前的K-Means算法并不是很稳定，20次实验中还出现了1次实验的结果是无效的情况. 改进后的算法准确率较为稳定，不会出现较大的波动. 基于Iris和Wine数据集所展开的一系列针对性实验可知，在对算法做出改进之后，有效避免了之前选择初始点的随机性现象，确保所选择初始点不会出现相临近的情况，成功克服了聚类结果局部最优的现象. 因而改进后的算法，无论是在准确率还是稳定性上均得到了一定程度的提高.

4 集群单机性能对比实验

4.1 单机串行和并行性能对比

实验比较了K-Means算法在一台机器上以串行方式运行所需的时间和使用在本文中配置的集群上进行并行运行的方式运行K-Means算法的时间. 本次实验所涉及到的数据都是人工数据，共40维，我们分别构造了50万条、200万条、400万条、500万条和600万条的数据集. 最后串行和并行的实验结果见表5.

表5 单机处理比较实验图

序号	文件大小/MB	记录数量/10^6条	串行时间t_1/ms	并行时间t_2/ms
1	57.3	0.5	50 098	120 082
2	229.7	2	201 492	196 834
3	459.4	4	423 891	310 234
4	574.3	5	684 669	364 012
5	689.2	6	outOfMem	410 023

在数据量较小的情况下，串行执行所需时间比集群执行所需的时间要少，这是因为 MapReduce 编程模型下的 K-Means 算法在迭代优化初始点的过程需要不断启动新的任务，这需要消耗一定的时间去准备. 每次的 MapReduce 程序执行过程均涉及到读取数据、移动数据、存储数据、网络传输等步骤，这会消耗大量时间，而真正花费在 K-Means 迭代上的时间并不多. 随着运行数据量的增加，并行的算法优势便逐渐体现出来. 单机串行的执行时间随着数据量的增加呈现指数增长趋势，而并行算法是呈现成比例增加的趋势.

4.2　加速比

算法的加速比公式如下所示：

$$\text{加速比} = \frac{\text{单机计算时间}}{\text{集群计算时间}}$$

在当就并行程序判断单机串行的性能时多引入这一概念. 实验结果如图 3 所示.

图 4 展示的实验中分别选择 1,2,3,4 个 TaskTracker 节点参与运算. TaskTracker 在每个 Slave 节点上都存在一个，在处理任务时，它可以开启一个或多个 JVM 来并行处理 map 和 reduce 任务.

实验结果表明，在数据量较小时，一个节点的加速比均表现不佳，这是因为单机处理少量数据比集群更加高效. 当数据量和节点数有所增加时，加速比会稳定增长，当处理更大的数据集时，能够有更好的加速比. 究其原因主要有两个方面：首先是对 MapReduce 做出了针对性的优化与改进，并在一定程度上压缩了网络传输及输入输出的数据量；其次由于数据量的递增，K-Means 处理所需的时间也按比例增加，并行程序的优势得到了更好的体现.

4.3　伸缩性

具有良好伸缩性的集群，应当是随着计算的数据集规模的增加，计算所需的时间按比例增加. 当集群节点数增加时，计算的效率得到了显著提高，比例也有所增加. 本文针对伸缩性的实验，采用了上一节相同的人工数据，分别在不同节点数执行不同规模的数据集，得到的实验结果如图 4 所示.

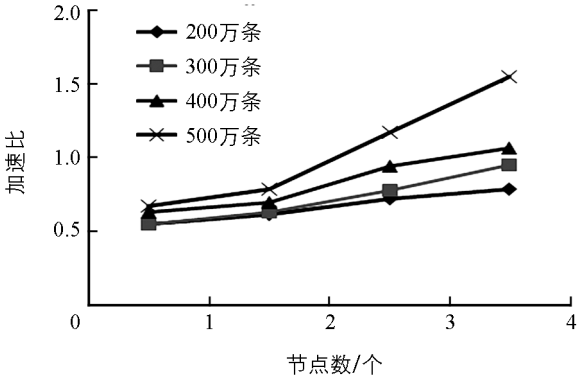

图 3　加速比测试结果图

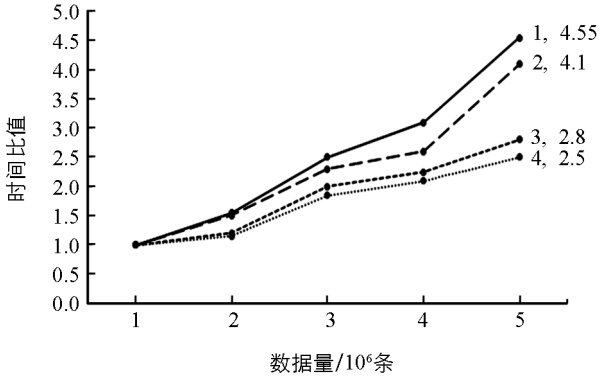

图 4　伸缩性测试结果图

集群在处理 100 万条记录的时间为单位 1，然后计算处理其他的数据量所消耗的时间与 100 万条记录处理时间的比值. 实验表明，随着数据集规模的增加，消耗的时间逐渐增加，节点越多，消耗的时间增加得更加缓慢. 例如，在 4 个节点情况下，处理 500 万条记录所需时间为处理 100 万条记录所需时间的 2.5 倍，但是在 1 个节点下，为 4.37 倍. 由此表明，并行 K-Means 算法适合运行在大规模集群中且能够有效地处理大规模的数据.

5　结束语

本文对 Hadoop 集群、K-Means 算法以及 K-Means 算法在 Hadoop 集群 MapReduce 模型上的实现进行了深入研究. 先是对 Hadoop 集群做出了概括性介绍，紧接着又简要介绍了聚类算法的相关知识，在此引入了基于划分的 K-Means 算法. 不仅系统阐述了这一算法的思想与实现过程，还通过分析探究了这一算法的优点与不足之处，然后针对 K-Means 算法的不足，对其进行了优化，并且将优化后的 K-Means 算法在 MapReduce 模型下成功实现.

参考文献:

[1] 江小平,李成华,向文,等. K-means聚类算法的MapReduce并行化实现[J]. 华中科技大学学报(自然科学版),2011,39(1):120-124.

[2] 周 锋,李旭伟. 一种改进的MapReduce并行编程模型[J]. 科协论坛,2009(2):65-66.

[3] CHANG F, DEAN J, GHEMAWAT S, et al. Bigtable: A Distributed Storage System for Structured Data [J]. ACM Transactions on Computer Systems, 2008, 26(2): 205-218.

[4] 翟东海,鱼 江,高 飞,等. 最大距离法选取初始簇中心的K-means文本聚类算法的研究[J]. 计算机应用研究,2014,31(3):713-715.

[5] SHVACHKO K, KUANG H, RADIA S, et al. The Hadoop Distributed File System [C]. 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies. New York: IEEE Press, 2010: 1-10.

[6] FRANK A. ASUNCION A. UCI Machine Learning Repository [EB/OL]. [2014-5-2]. http://archive.ics.uci.edu/ml/.

[7] 万川梅. 深入云计算:Hadoop应用开发实战详解[M]. 北京:中国铁道出版社,2013.

[8] 陆嘉恒. Hadoop实战[M]. 北京:机械工业出版社,2012.

[9] 冯登国,张 敏,张 妍,等. 云计算安全研究[J]. 软件学报,2011,22(1):71-83.

[10] DEAN J, GHERNAWAT S. MapReduce: Simplified Data Processing on Large Clusters [J]. Operating Systems Design and Implementation, 2008, 55(1): 107-113.

[11] 周 涓,熊忠阳,张玉芳,等. 基于最大最小距离法的多中心聚类算法[J]. 计算机应用,2006,26(6):1425-1427.

[12] 黄 山,王波涛,王国仁,等. MapReduce优化技术综述[J]. 计算机科学与探索,2013,7(10):865-885.

[13] LOMBARDI F, PIETRO R D. Secure Virtualization for Cloud Computing [J]. Journal of Network and Computer Applications, 2011(34): 1113-1122.

MapReduce Parallel Implementation of K-Means Algorithm

JIANG Yi, LIU Xin-yang

College of computer science and technology, Chongqing University of Posts and Telecommunications, Chongqing 40065, China

Abstract: This paper firstly built a Hadoop cluster, then it realizes K-Means algorithm in the MapReduce model, and finally it designs multigroup experiments to verify that the K-Means algorithm has better stability and accuracy for data clustering analysis.

Key words: Hadoop Distributed; K-Means; MapReduce

责任编辑 张 枸