一种K-means改进算法的并行化实现与应用

李晓瑜[1]，俞丽颖[1]，雷 航[1]，唐雪飞[1,2]

(1. 电子科技大学信息与软件工程学院 成都 610054；2. 成都康赛信息技术有限公司 成都 610054)

【摘要】随着数据的爆炸式增长，聚类研究作为大数据的核心问题之一，正面临计算复杂度高和计算能力不足等诸多问题。提出了一种基于Hadoop的分布式改进K-means算法，该算法通过引入Canopy算法初始化K-means算法的聚类中心，克服传统K-means算法因初始中心点的不确定性，易陷入局部最优解的问题。本算法在Canopy(罩盖)中完成K-means聚类，并在Canopy间完成簇的合并，聚类效果稳定，迭代次数少。同时，结合MapReduce分布式计算模型，给出改进后算法的并行化设计方法和策略，进一步通过改进相似度度量方法，将该方法用于文本聚类中。实验结果证明该算法具有良好的准确率和扩展性。

关 键 词 canopy算法； Hadoop； MapReduce； 并行K-means； 文本聚类

中图分类号 TP311 文献标志码 A doi:10.3969/j.issn.1001-0548.2017.01.010

The Parallel Implementation and Application of an Improved K-means Algorithm

LI Xiao-yu[1], YU Li-ying[1], LEI Hang[1], and TANG Xue-fei[1,2]

(1.School of Information and Software Engineering, University of Electronic Science and Technology of China Chengdu 610054;
2. Chengdu COMSYS Information Tech. Co., Ltd Chengdu 610054)

Abstract Following with the growth of massive data, clustering research, one of the core problems of big dataisfaced with more and more problems such as high computing complexity and lack of resource. It has proposed an improved parallel K-means algorithm based on Hadoop. To overcomethe problem that the traditional K-means algorithm often has local optimal solution due to the randomness choice of initial center, we introduce Canopy algorithm to initialize clustering center andapply K-means algorithm on canopy. Meanwhile, clusters are merged among canopies. The result is stable and iteration number is less. In addition, the parallel implementation methods and strategies of the improved algorithm are presented, combining with the distributed computing model of MapReduce. And a new method of text clustering is introduced by improving the similarity of measurement. The experiment results indicate the validity and scalability of our method.

Key words canopy algorithm; Hadoop; MapReduce; parallel K-means; text clustering

随着计算机和存储技术的快速发展，在商业、社会、工程和医学等各方面都会产生大规模的数据，人们开始关注如何对大规模的海量数据进行分析和科学研究，进而辅助商业决策和企业管理，高效地发现隐藏在数据中的有用知识。因此，对海量数据的挖掘得到了广泛的研究和关注。

聚类分析是数据挖掘领域最重要的研究方向之一。“物以类聚、人以群分”，聚类算法是将物理或抽象的对象分成相似对象集合的过程。簇是数据对象的集合，同一簇中的对象彼此相似，而与其他簇中的对象相异[1-2]。与其他数据挖掘方法相比，聚类不需要先验知识，就可以完成数据的分类。聚类算法可以分为基于划分的、密度的、模型的等多种类型[3]。

在基于划分的聚类算法中，K-means算法被广泛使用，它具有算法数学思想简单、收敛速度快且易于实现等多种优点[4]，但存在需要事先制定聚类个数，以及由于中心点选择的随机性而易陷入局部最优解的问题。随着数据量的增大，传统的K-means算法在对海量数据集进行分析时，已经很难满足现实需要。针对传统K-means算法的缺点，已有很多学者在K-means的基础上提出了改进措施。文献[5]针对初始聚类中心选择的问题，提出了一种基于最优划分的聚类中心选择算法，该算法通过对数据集进行初始划分，确定K-means的初始中心，提高了聚类

收稿日期：2015－06－03；修回日期：2015－12－09
基金项目：国家科技支撑计划(2012BAH87F03)；中央高校基本科研业务费(ZYGX2014J065)
作者简介：李晓瑜(1984－)，女，博士，主要从事大数据分析与应用、量子计算和量子信息等方面的研究.

的准确度，但算法的递归次数会随数据样本维度的的增加而激增，因此导致算法实时性降低。文献[6]提出了一种通过采样和K-means预聚类，构造相交子簇的加权连通图，进而合并子簇得到最终聚类结果的改进K-means算法，该算法提高了K-means算法局部聚类的精度，但由于其缺乏对样本空间的整体把握，聚类效果仍有待提高。文献[7]提出使用Canopy算法优化K-means算法，进一步优化了初始中心的选择问题，但Canopy算法初始阈值大小的确定一般靠人工选取，因此效果并不稳定。此外，文献[8]采用AP聚类算法[9]来确定 k 可取的最大值；文献[3]提出采用最大最小距离法来选取初始聚类中心。基于以上算法本文以传统的K-means聚类算法为基础，探讨了如何在MapReduce分布式框架下，快速、准确、高效地进行聚类，提出了一种针对海量数据挖掘的分布式聚类算法：即通过Canopy算法初始化来选择K-means中心点，使待聚类的每个点在其所属Canopy中进行聚类，重新计算中心点，同时进行邻近Canopy的合并，反复以上过程直至收敛。并通过使用余弦相似度算法，将方法用于文本聚类中。

1 相关背景

1.1 并行计算理论

并行计算是指同时使用多种计算资源解决计算问题的过程。其目的是快速解决大型且复杂的计算问题。其首要任务是将一个应用分解成多个子任务，将其分配给不同的处理器，同时各个处理器之间相互协同，并行地执行子任务，最后将所有子任务的结果合并形成整体的计算任务结果，从而达到加速求解的目的。

Hadoop是Apache软件基金会旗下的一个开源分布式计算平台，以Hadoop分布式文件系统(HDFS)和MapReduce(Google MapReduce的开源实现)为核心的Hadoop，为用户提供了系统底层细节透明的分布式基础框架[10]。MapReduce是一种用于处理海量数据的编程模型，它将并行处理海量数据的过程抽象成为“Map(映射)”和“Reduce(化简)”两个步骤[11]。Map任务将输入的文件转换成一个key-value对序列，Reduce将Map输出的键值对序列以某种方式进行合并和汇总，同时，Reduce输出的key-value对会按照key位进行排序。MapReduce的处理流程如图1所示。MapReduce还提供Combine函数，Combine函数在Map结束后于本地进行结果合并，相当于本地的Reduce，可以大大减少网络I/O操作的消耗。通过MapReduce提供的接口，使得开发人员大大减少了并行化开发的工作量，可以高效地利用分布式资源，而不需要分布式计算的开发经验[12]，实现该框架下的应用程序可以运行在大规模集群，而且集群对应用程序是透明的，具有很好的容错机制，能够可靠地并行处理大规模数据。

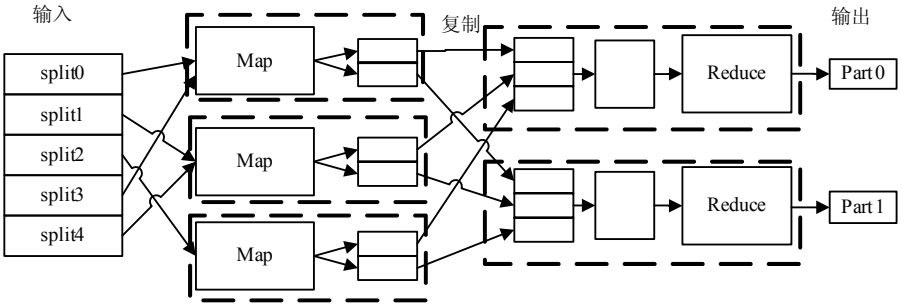

图1 MapReduce处理流程

1.2 聚类算法

聚类是数据挖掘中的重要算法之一，它的目的是在无先验知识的条件下，按照数据集中数据本身的特点和差异，把一个数据集分割成若干个不同的类或簇，使得在同一个类中的数据对象的差异性尽可能小，而处于不同类中的数据对象的差异性尽可能的大。即可将聚类定义为：若数据集 X 中有n个数据点 x，其中 $X=(x_1,x_2,\cdots,x_n)$，聚类的最终目的就是把数据集 X 分为 k 个类 C_i，使得 $X=C_1\cup C_2\cup\cdots\cup C_k\cup C_n$ (其中，C_n 为噪声)且 $C_i\cap C_j=\Phi$ ($i\neq j$)。需要说明的是，在模糊聚类中，每个数据点可能会属于多个类，因此类间可能会存在交集。

K-means算法是聚类算法中使用最为广泛的算法之一，其用到的数学思想简单，但效率高，且在对“圆形−球形”性质集合进行分类时，可以达到良好的聚类结果。传统K-means算法的执行过程如下所述[13]：

1) 针对数据集 $X=(x_1,x_2,\cdots,x_n)$ 首先通过人工指定数字K作为要聚类的数目，并随机从样本点中选取相应数目的初始聚类中心 $\{c_1,c_2,\cdots,c_k\}$；

2) 计算各个样本点 x_i 到这 k 个聚类中心的距离，并把样本 x_i 归到与他距离最近的那个聚类中心 c_i 所在的类中；

3) 根据前一步的结果，计算同一个类中所有样本点的各维平均值，作为新一轮聚类的 k 个中心；

4) 再重复上述过程继续聚类，直至收敛。

简单地说，K-means聚类的目的，就是使得式(1)最小化：

$$J=\sum_{i=1}^{n}\sum_{j=1}^{k}r_{ij}\left\|x_i-c_j\right\|^2 \tag{1}$$

其中，

$$r_{ij}=\begin{cases}1 & x_i\in c_j\\ 0 & x_i\notin c_j\end{cases} \tag{2}$$

K-means算法简单高效，但其缺点也十分明显。首先，聚类的数目依赖于 k 值设定，而在实际中，k 值一般是难以界定的，而 k 值的指定决定了聚类的结果。其次，初始聚类中心是随机选择的，使得最终的聚类结果依赖于初始中心点的选择，导致结果的不稳定性[14]。同时，在标准K-means算法中，许多距离计算是冗余的，因此当计算量很大时，算法的时间开销也非常大。

Canopy[15]算法也是一种聚类方法，主要用于海量高维数据的聚类。Canopy可以粗略地将数据划分成若干个重叠子集，即Canopy(罩盖)。每个子集作为一个类簇，其一般使用一种代价低的相似性度量方法，以加快聚类的速度[16]。因此，Canopy聚类一般用于其他聚类算法的初始化操作。Canopy算法中，Canopy(罩盖)的形成需要指定两个距离阈值，T_1、T_2 ($T_1>T_2$)。首先，在数据集中选择一个点作为初始中心点加入到Canopy(罩盖)中心列表 C 中。对数据集中任意一个点 x_i ($x_i\notin C$)，若 x_i 与 c_j ($c_j\in C$)的距离均大于 T_1，则将 x_i 作为一个新的Canopy(罩盖)中心加入到 C 中；若距离小于 T_1，则将 x_i 加入以 c_j 为中心的罩盖中；若 x_i 与 c_j 的距离小于 T_2，将 x_i 与 c_j 强关联，x_i 不再能作为其他罩盖的中心，因此将其从数据集中删去；反复以上过程，直至数据集为空[17]。Canopy聚类允许有重叠子集，增加了算法的容错性和消除孤立点作用；同时，由于只需在罩盖内进行精确聚类，从而避免了对所有点精确聚类带来的计算量大的问题。

2　改进的Canopy-K-means算法

2.1　算法思想

传统的K-means算法由于初始聚类中心选择的随机性，算法结果随着中心点选择的不同而改变，导致结果的不稳定性，可能会造成局部最优值的问题[18]。针对中心点选择的问题，将Canopy算法作为传统K-means算法的初始化操作，快速地找出相对准确的聚类中心。本文将每一次Canopy形成的重叠子集称为罩盖，而每次K-means聚类形成的不重叠子集称为簇。在Canopy算法初始化过程中生成的重叠罩盖，使得数据集中的每一个点都至少属于一个罩盖中，因此，在进行K-means聚类的过程中，本文算法将不再考虑每个点到所有中心的距离，而只需计算点到其所属罩盖中心的距离，简而言之，K-means算法将在每个罩盖中进行，而随着K-means算法的迭代，每个罩盖中心也将不断变化，直至收敛。具体执行流程如图2所示。

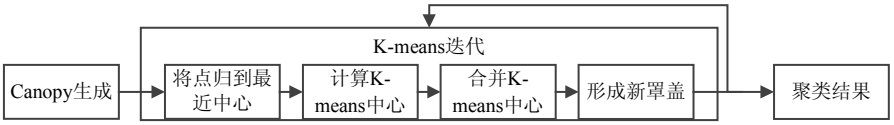

图2　改进Canopy-K-means算法执行流程

由图2可知，改进后的Canopy-K-means算法首先通过Canopy算法形成相互重叠的罩盖，再进行K-means迭代，最后产出聚类结果。具体分为6步：

1) 生成Canopy。即对于数据集 $X=(x_1,x_2,\cdots,x_n)$，通过Canopy算法指定的阈值 T_1，T_2，迭代地找出初始中心点 $\{c_1,c_2,\cdots,c_k\}$，从而形成相互重叠的Canopy(罩盖)。

2) 数据点划分。在生成Canopy之后，由于数据集中每个数据点至少属于一个Canopy(罩盖)，而属于不同罩盖的点一定不属于同一个簇。因此，在计算新中心前，每个点已经明确属于某一个或几个罩盖中，从而不需要再计算每个点与每个中心点的距离，而只需计算其和其所属罩盖中心点的距离，并将其归入到其距离最近的罩盖中心点所属的Canopy(罩盖)中，形成相互不重叠的簇，减少了计算量。即对于任意点 x_i，假设 $Cn_{x_i}=\{c_1,c_2,\cdots\}$ 是其所属Canopy中心集合，若 x_i 与其所属的某一个Canopy的中心点 c_j 距离最小，则将 x_i 划到 c_j 所属的Canopy中，并将其从其他Canopy中删除。

3) 计算K-means中心。利用上一步生成的簇，

计算每个簇的新中心点，即求平均的标准K-means算法。

4) 合并K-means中心。由于前期采用了Canopy算法生成初始中心，而初始半径主要靠人工确定，如果初始半径选择过小，可能会出现中心点较多，或是在进行K-means聚类之后，新的中心点比较靠近的情况，因此在新中心点产生之后，对新中心进行检查，将距离较近的中心点进行合并，其对应所处的簇也相应合并，并计算合并后的新中心，从而产生一次迭代的最终K中心点，以避免人工确定半径带来的聚类效果不稳定的问题。

5) 形成新簇，产生新的重叠罩盖，并迭代。计算合并后K中心落在上一步产生的哪些Canopy(罩盖)中，将这些Canopy(罩盖)的中心点替换为新的K中心，形成新的重叠Canopy(罩盖)。该步骤和步骤4)就是合并产生新Canopy的过程。从步骤2)开始，重复以上步骤，直至算法收敛。

6) 形成聚类结果。在迭代结束之后，得到最终的罩盖，由于算法已经收敛，此时，只需将属于多个罩盖中的点归入到其距离最近的中心点所在Canopy(罩盖)中，形成不重叠的最终簇，完成聚类。

2.2 基于MapReduce改进的Canopy-K-means算法的并行化设计

由上面的分析可知，本算法主要由3个阶段组成，即Canopy初始化阶段、K-means迭代聚类阶段，以及聚类结果产生阶段。

1) Canopy初始化阶段

在基于MapReduce的并行化算法中，Canopy初始化阶段有分为两个步骤：Canopy中心点的产生和Canopy的形成，分别对应于两个Job，分别为Job1和Job2，两个Job的执行流程及Map和Reduce过程实现如图3所示。

Job1中Map函数产生局部数据集的Canopy中心，Reduce阶段收集局部中心点，并执行与Map阶段同样的操作，合并局部中心点集，最后得到全局中心点集，并以<centorid，中心点各维度值>形式存放在本地，MapReduce过程可表示为：

$$\text{Map：}\ X,T_2 \rightarrow <\text{"centroid"},c_i>$$

其中，X 为原始数据集，T_2 为阈值，"centroid"为map的key值，c_i 为计算得到的初始中心点值，$c_i \in C_n$，Reduce过程与其类似；

Job2在Map阶段，每个Map节点加载Job1产生的全局Canopy中心，遍历子集中的所有数据点，判断其与Canopy中心的距离，将其加入与其距离小于t_1的Canopy，Reduce阶段负责整理各个Map产生的结果，产生Canopy，以<数据ID：数据各维值，所属Canopy列表(Canopy中心各维值)>的key-value对形式存储，MapReduce过程可表示为：

$$\text{Map：}\ X,T_1,C_n \rightarrow <i:x_i,c_j>$$

$$\text{Reduce：}\ <i:x_i,c_j> \rightarrow <i:x_i,\text{List}(c_j)>$$

其中，C_n 为所有Canopy中心的集合，$\text{List}(c_j)$ 为相应数据点所属Canopy中心集合，$c_j \in C_n$。

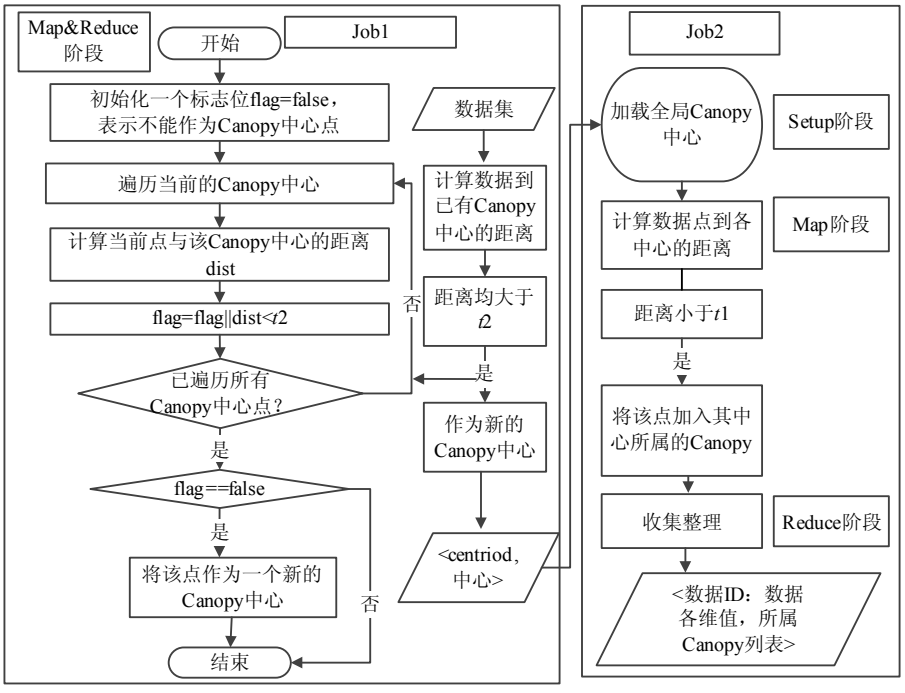

图3 Canopy初始化阶段流程

2) K-means迭代聚类阶段

在K-means迭代聚类阶段，在MapReduce框架下可以分为两个步骤：第一步是利用Canopy中每个点各维度的值，重新计算每个Canopy中心；第二步合并邻近的新中心，并判断其属于哪个Canopy，将属于该Canopy的点的中心替换成为新的合并后的中心值，形成新的重叠Canopy。因此这一阶段也是通过两个Job完成，分别对应于Job3和Job4，两个Job的执行流程及Map和Reduce过程实现如图4所示。

Job3中的Map函数接收Job2产生的<数据ID：数据各维值，所属Canopy列表(Canopy中心各维值)>key-value对，计算与每个点距离最近的Canopy中心，将其加入该Canopy中，形成<簇，数据ID：数据各维值>；Map函数的输出在本地通过Combine函数，实现同一簇数据对象的合并，并统计数据的个数，得到<簇，数据点各维和+数据个数>，由于只能输出key-value对的形式，在实现中，将数据个数作为各维度和的最后一位加入到value中；Reduce函数接收个Combine函数的输出，进行汇总，得到各个簇的所有点各维度之和以及各簇的数据点总数，最终得到每个簇的新中心K，输出<簇，新K中心各维值>；这里Combine函数的设计，有效地完成了局部数据的统计，减少了节点间的通信代价。

Job3的MapReduce过程可表示为：

Map：$< i : x_i, \mathrm{List}(c_j) > \rightarrow < c_j, i : x_i >$

Combine：$< c_j, i : x_i > \rightarrow < c_j, \sum x_i : n_j >$

Reduce：$< c_j, \sum x_i : n_j > \rightarrow < c_j, c_j' >$

其中，n_j 表示经过将每个数据点归类到其最近Canopy后，属于集合 C_{nj} 的数据对象个数，c_j' 为K-means计算后得到的新中心的值。

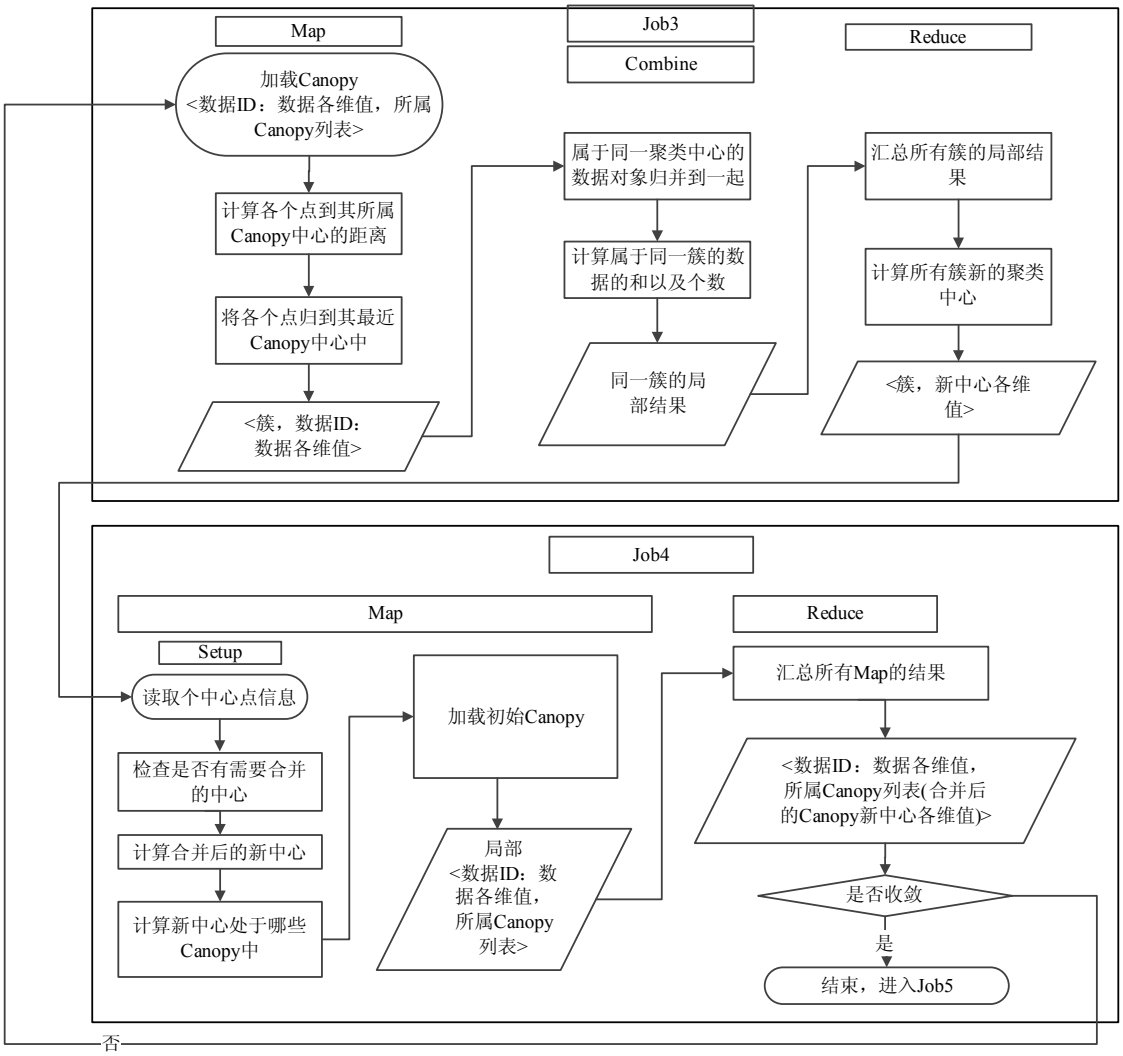

图4　K-means迭代聚类阶段流程

Job4接收Job3的输出，首先在各个Map节点的初始化(Setup)阶段，检查是否有需要合并的K中心，如

果有，将其进行合并，合并后产生得到的新中心，判断其属于那些Canopy中，得到合并后的K中心与原初始的Canopy的对应关系，判断其是否收敛。在Map阶段，将所有数据点的所属的Canopy列表中心替换为合并后的新中心，之后Reduce函数进行汇总，最终形成形如<数据ID：数据各维值，所属Canopy列表(合并后的Canopy新中心各维值)>。在Setup时进行的收敛判断中，若收敛，则在Map完成之后直接进入Job5以产生最后结果，若不收敛，则从Job3开始循环直至收敛。

Job4的MapReduce过程可表示为：

Map：

$$<c_j,c_j'>,T_2,<i:x_i,\mathrm{List}(c_j)>\rightarrow<i:x_i,\mathrm{List}(c_j'')>$$

其中，c_j''是合并后新中心的值，Reduce过程类似。

3) 聚类结果产生阶段

聚类结果产生阶段对应于Job5，接收Job4中产生的各个数据点与一个或多个稳定的K中心对应关系，在Map阶段将局部数据点归到与其距离最近的K中心，输出<数据ID：数据各维值，所属簇(最终K中心各维)>，完成聚类，该过程与Job3的map过程类似。

3 改进后Canopy-K-means算法的文本聚类

3.1 文本预处理

利用本文提出的改进Canopy-K-means算法进行文本聚类，采用文本向量空间模型VSM对文本进行预处理。即给定文本集 $D=\{d_1,d_2,\cdots,d_n\}$，d_i 表示每个文本向量，且 $d_i=(<t_1,w_{i1}>,<t_2,w_{i2}>,\cdots,<t_j,w_{ij}>)$。其中 $T=\{t_1,t_2,\cdots,t_j\}$ 表示从所有文本中提取的特征词集合，$W_i=\{w_{i1},w_{i2},\cdots,w_{ij}\}$ 表示文本 d_i 中包含各个特征词所对应的权重。向量权重的计算采用统计方法TF-IDF(term frequency-inverse document frequency)[3]来计算：

$$w_{ij}=f_{ij}\log(N/n_i+0.01) \tag{3}$$

式中，w_{ij} 表示词语 i 在文档 j 中的权重值；f_{ij} 为词语 i 在文档 j 中的出现次数；n_i 表示整个文本集中出现过词语 i 的文本数；N 为文档总数。由式(3)可以看出，词语 i 在该文本中出现的次数越多，其权值越大；而整个文本集中出现过词语 i 的文本越多，其权值则会相应减少，也就是说，字词的权重与它在文件中出现的次数成正比，与它在资料库中出现的频率成反比[10]。具体实现中，在构建向量空间模型之前，采用中科院计算机研究所研制的汉语词法分析系统NLPIR[19]，同时，利用哈工大中文停词表去除广泛使用词语和实际意义很小的词语，最后选择特征较大的词语形成VSM中的维度。

3.2 文本距离计算

本文算法使用两个点之间的距离作为衡量两个点是否相似的标准。而在文本聚类中通常是使用两个文本向量间夹角的余弦值，即余弦相似度，来衡量两个文档的相似程度。而由于两个文本之间，向量夹角余弦值越大，相似度越大，说明其距离越小，也就是相似度和距离成反比，因此构造式(4)来定义两个文本 d_i、d_j 之间的距离。

$$\mathrm{dist}(d_i,d_j)=\log_a(\mathrm{sim}(d_i,d_j))) \quad (a\in(0,1)) \tag{4}$$

式中，

$$\mathrm{sim}(d_i,d_j)=\frac{d_i d_j}{|d_i||d_j|} \tag{5}$$

式中，$\mathrm{dist}(d_i,d_j)$ 为两个文本的距离；$\mathrm{sim}(d_i,d_j)$ 为两个文本的余弦相似度。

4 实验结果与分析

4.1 实验数据及预处理

实验采用UCI Machine Learning Repository[20]机器学习数据集中的Iris和Wine数据集，其中，Iris数据集共有150个样本，每个样本4个属性，共分为3类；Wine数据集共178个样本，每个样本13个属性，分为3类。文本聚类方面，选用搜狗实验室文本分类语料库[21]中选取财经、体育、旅游、教育4个类别的中文新闻文档各1 000篇。另外，为了对算法的扩展性进行验证，将Iris数据集构造成维度60维，行数为100万行(0.25 G)、300万行(0.6 G)、500万行(1.1 G)、900万行(2 G)的更大规模数据集。

4.2 实验结果

实验在由4台Ubuntu14.04平台下搭建的Hadoop集群下完成，由1台作为namenode，3台作为datanode，其配置均为2 GB内存，80 G硬盘，Hadoop版本为1.2.1。

1) 算法准确度

利用Iris数据集、wine数据集对算法聚类准确性进行检验，实验一共进行20次，结果为这20次的平均值。在传统K-means过程中，由于每次选择的初始中心不同，聚类效果不稳定，迭代次数较多，而本文算法由于优化了中心的选择，因此结果十分稳定。由表1可知，本文算法的正确率高于传统K-means算法，而误差平方和以及迭代次数比K-means算法更

低，说明本文算法聚类结果的类内相似度更高且收敛速度更快。

表1　数据集测试结果

数据集	传统K-means			本文算法		
	正确率	误差平方和	迭代次数	正确率	误差平方和	迭代次数
Iris	0.842	92.56	10	0.903	78.95	5
Wine	0.887	48.98	8	0.921	41.94	6

在利用本文算法进行文本聚类时，采用前文中提到的通过文本间余弦相似度来计算距离的方法来衡量文本间的距离，在不同的单词提取率下比较本文算法与传统K-means算法在文本聚类上的准确率(precision)、召回率(recall)和*F*度量值[22]，如表2所示。

可以看到，本文算法可以很好地应用到文本聚类中。

表2　文本聚类测试结果

单词提取率	传统K-means			本文算法		
	准确率	召回率	*F*度量	准确率	召回率	*F*度量
0.002	0.61	0.62	0.62	0.76	0.74	0.75
0.005	0.75	0.68	0.72	0.87	0.73	0.80
0.01	0.68	0.66	0.67	0.81	0.70	0.75

2) 算法扩展性

为了分析算法在Hadoop框架下并行执行的性能，需要计算算法执行的加速比。对同一个计算任务，并行算法的加速比等于单机执行时间除以并行执行时间的值。加速比是用来衡量程序执行并行化的重要指标[13,15]。算法加速比曲线如图4所示，由图4可知，本文算法在并行化后有良好的加速比，并随着数据集的增大效果更佳明显。

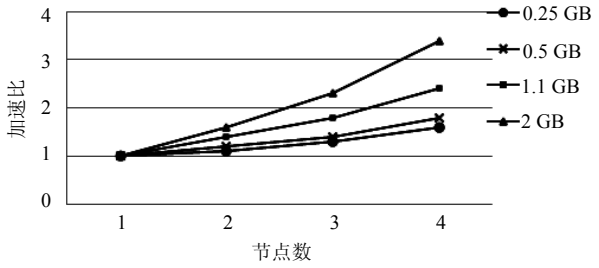

图4　Hadoop下并行加速比

5　结 束 语

本文以海量数据聚类为背景，改进了原有传统K-means算法，克服了其随机选择中心点带来的结果不稳定问题，提高了聚类结果的准确性和稳定性，减少了聚类次数；并利用MapReduce框架给出了算法的并行化设计。同时结合实际文本聚类应用场景，给出了本算法在文本聚类方面的应用。实验验证了本文算法具有良好的有效性和扩展性。

参 考 文 献

[1] 孙吉贵, 刘杰, 赵连宇. 聚类算法研究[J]. 软件学报, 2008, 19(1): 48-61.
SUN Ji-gui, LIU Jie, ZHAO Lian-yu. Clustering algorithm research[J]. Journal of Software, 2008, 19(1): 48-61.

[2] JAIN A K, MURTY M N, FLYNN P J. Data clustering: a review[J]. ACM Computing Surveys (CSUR), 1999, 31(3): 264-323.

[3] 翟东海, 鱼江, 高飞, 等. 最大距离法选取初始聚类中心的K-means文本聚类算法的研究[J]. 计算机应用研究, 2014, 31(3): 713-715, 719.
ZHAI Dong-hai, YU Jiang, GAO Fei, et al. K-means text clustering algorithm based on centers selection according to maximum distance[J]. Application Research of Computers, 2014, 31(3): 713-715, 719.

[4] 赵庆. 基于Hadoop平台下的Canopy-Kmeans高效算法[J]. 电子科技, 2014, 27(2): 29-31.
ZHAO Qing. Efficient algorithm of canopy-kmeans based on Hadoop platform[J]. Electronic Science and Technology, 2014, 27(2): 29-31.

[5] 张健沛, 杨悦, 杨静, 等. 基于最优划分的K-Means初始聚类中心选取算法[J]. 系统仿真学报, 2009, 21(9): 2586-2589.
ZHANG Jian-pei, YANG Yue, YANG Jing, et al. Algorithm for initialization of K-means clustering center based on optimized-division[J]. Journal of System Simulation, 2009, 21(9): 2586-2589.

[6] 雷小峰, 谢昆青, 林帆, 等. 一种基于K-means局部最优性的高效聚类算法[J]. 软件学报, 2008, 19(7): 1683-1692.
LEI Xiao-feng, XIE Kun-qing, LIN Fan, et al. An efficient clustering algorithm based on local optimality of K-means[J]. Journal of Software, 2008, 19(7): 1683-1692.

[7] 邱荣太. 基于Canopy的K-means多核算法[J]. 微计算机信息, 2012(9): 486-487.
QIU Rong-tai. Canopy for K-means on multi-core[J]. Microcomputer Information, 2012(9): 486-487.

[8] 周世兵, 徐振源, 唐旭清. 新的K-均值算法最佳聚类数确定方法[J]. 计算机工程与应用, 2010, 46(16): 27-31.
ZHOU Shi-bing, XU Zhen-yuan, TANG Xu-qing. New method for determining optimal number of clusters in K-means clustering algorithm[J]. Computer Engineering and Applications, 2010, 46 (16): 27-31.

[9] FREY B J, DUECK D. Clustering by passing message between data points[J]. Science, 2007, 315: 972-976.

[10] 陆嘉恒. Hadoop实战[M]. 2版. 北京: 机械工业出版社, 2012.
LU Jia-heng. Hadoop in action[M]. 2nd ed. Beijing: China Machine Press, 2012.

[11] DEAN J, GHEMAWAT S. MapReduce: Simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1): 107-113.

[12] 丁智, 林治. MapReduce编程模型、方法及应用综述[J]. 电脑知识与技术, 2014, 10(30): 7060-7064.

DING Zhi, LIN Zhi. Review on MapReduce programming model, method and application[J]. Computer Knowledge and Technology, 2014, 10(30): 7060-7064.

[13] 陈爱平. 基于Hadoop的聚类算法并行化分析及应用研究[D]. 成都: 电子科技大学, 2012.
CHEN Ai-ping. The parallel analysis and application research on clustering algorithm based on Hadoop[D]. Chengdu: University of Electronic Science and Technology of China, 2012.

[14] 韩凌波, 王强, 蒋正锋, 等. 一种改进的K-Means初始聚类中心选取算法[J]. 计算机工程与应用, 2010, 46(17): 150-152.
HAN Ling-bo, WANG Qiang, JIANG Zheng-feng, et al. Improved K-means initial clustering center selection algorithm[J]. Computer Engineering and Applications, 2010, 46(17): 150-152.

[15] ESTEVES K M, RONG C. Using Mahout for clustering Wikipedia's latest articles: a comparison between K-means and fuzzy c-means in the cloud[C]//Proceedings of the 2011 Third IEEE International Conference Science, Cloud Computing Technology and IEEE Computer Society. Washington, DC, USA: IEEE, 2011: 565-569.

[16] 余长俊, 张燃. 云环境下基于Canopy聚类的FCM算法研究[J]. 计算机科学, 2014, 41(11A): 316-319.
YU Chang-jun, ZHANG Ran. Research of FCM algorithm based on canopy clustering algorithm under cloud environment[J]. Computer Science, 2014, 41(11A): 316-319.

[17] MCCALLUM A, NIGAM K, UNGAR I H. Efficient clustering of high-dimensional data sets with application to reference matching[C]//Proceedings of the Sixth ACM SIUKDD International Conference on Knowledge Discovery and Data Mining. [S.l.]: ACM, 2000: 169-178.

[18] 樊宁. K均值聚类算法在银行客户细分中的研究[J]. 计算机仿真, 2011, 28(3): 369-372.
FAN Ning. Simulation study on commercial bank custermer segmentation on K-means clustering algorithm[J]. Computer Simulation, 2011, 28(3): 369-372.

[19] 张华平. 自然语言处理与信息检索共享平台[EB/OL]. [2015-03-30]. http://www.nlpir.org/.
ZHANG Hua-ping. Natural language processing and information retrieval sharing platform[EB/OL]. [2015-03-30]. http://www.nlpir.org/.

[20] UCI. UCI Machine learning repository[DB/OL]. [2015-03-30]. http://archive.ics.uci.edu/ml/.

[21] 搜狗.文本分类语料库[DB/OL]. [2015-03-30]. http://www.sogou.com/labs/dl/c.html.
Sougou. Text classify lab data[DB/OL]. [2015-03-30]. http://www.sogou.com/labs/dl/c.html.

[22] YANG Y. An evaluation of statistical approaches to text categorization[J]. Information Retrieval, 1999, 1(1-2): 69-90.

编 辑 蒋 晓