# 云环境下基于数据流的k-means聚类算法

王 飞 秦小麟 刘 亮 沈 尧
(南京航空航天大学计算机科学与技术学院 南京 210016)

**摘 要** k-means算法是一种最常用的基于划分的聚类算法。传统的集中式k-means算法已不能适应当前呈爆炸式增长的数据规模,设计分布式k-means算法成为了目前亟需解决的问题。现有分布式k-means算法基于MapReduce计算框架且没有考虑初始聚类中心的影响。由于每个MapReduce任务均需要读写分布式文件系统,导致MapReduce不能有效表达多个任务之间的依赖关系,因此提出了一种基于数据流的计算框架,该框架建立在MapReduce之上,将数据处理过程按照数据流图建模。在该框架的基础上,提出了一种高效的k-means算法,它采用基于多次采样的初始聚类中心选取方法来实现负载均衡及减少迭代次数。实验结果表明,该算法的可扩展性较好,且效率比现有算法高。

**关键词** k-means,MapReduce,计算框架,数据流
**中图法分类号** TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.11.048

## Algorithm for k-means Based on Data Stream in Cloud Computing

WANG Fei QIN Xiao-lin LIU Liang SHEN Yao
(College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

**Abstract** k-means algorithm is one of the most commonly used clustering algorithm. Now data scale is exploding, and traditional centralized algorithm can not meet the requirements, so it is an urgent problem to design distributed k-means clustering algorithm currently. Existing distributed k-means algorithms are based on MapReduce framework and don't consider the clustering center. Since each MapReduce job reads and writes data from distributed file system, it is inefficient to express dependencies between jobs. Then this paper proposed a framework based on data stream. Based on MapReduce framework, this framework models according to the data flow diagram. And it proposed an efficient k-means algorithm on the framework. It uses an improved algorithm based on sampling to confirm clustering center for load balance and reducing iterations. Experimental results demonstrate that the algorithm can efficiently resolve the large scale k-means cluster.

**Keywords** k-means, MapReduce, Framework, Data stream

## 1 引言

聚类分析[1]是计算机科学及相关领域中的一个基础问题,在数据挖掘、模式识别、网络、生物信息等许多领域[2]有着广泛的研究和应用。聚类分析就是按照某个特定标准(如距离准则)把一个数据集分割成不同的类,使得同一个类里的数据对象之间的相似性尽可能的大,同时不在同一个类里的数据对象之间的差异性尽可能的大。即聚类之后,使得同一类的数据尽量聚集到一起,不同类的数据尽量分离。主要的聚类算法可分为基于密度、基于划分、基于网格、基于层次以及基于模型这5类。k-means[3](均值)聚类算法是一种应用最广泛的基于划分的聚类算法。该算法由于效率高,因此在对大规模数据进行聚类时被广泛应用。

在许多应用中需要对网页按照其内容进行聚类,或者对用户按照其行为进行聚类。例如,在社交网络中将具有某种关系准则联系在一起的人们划分为一个群体,然后可以针对这个群体预测其购买行为,设计针对性的营销方案等。随着互联网的飞速发展和数据的急剧增长,网络图以及社交网络图变得越来越大,例如,网络图可能包含数万亿条边[4]。聚类计算任务所面临的数据规模越来越庞大,以至于单个节点无法满足数据存储,基于单节点单进程的串行执行方法[5,6]已不可用。因此,设计云环境下的分布式k-means算法成为了目前亟需解决的问题。MapReduce[7]是Google提出的一种在集群上处理大规模数据集的分布式并行编程模型,也是目前云计算环境中的核心计算模式。为了有效地解决大规模数据集的k-means聚类问题,本文首先提出了基于MapReduce

到稿日期:2014-11-04 返修日期:2015-01-06 本文受国家自然科学基金项目(61373015,61300052),国家教育部高等学校博士学科点专项科研基金资助项目(20103218110017),江苏高校优势学科建设工程资助项目(PAPD),中央高校基本科研业务费专项项目(NP2013307),云计算-南航-大数据处理引擎技术研究项目资助。

王 飞(1989—),男,硕士生,主要研究方向为云环境下数据查询处理,E-mail:wangfnuaa@163.com;秦小麟(1953—),男,教授,博士生导师,主要研究方向为分布式数据管理与安全、信息安全等;刘 亮(1985—),男,博士后,讲师,主要研究方向为传感器网络数据库、流数据库等;沈 尧(1986—),男,博士生,主要研究方向为云计算。

框架的 k-means 算法。串行算法中的一次迭代就对应了一个 MapReduce 任务。

由于在现有的 MapReduce 框架开源实现版 Hadoop[8] 平台下，每个 MapReduce 任务都需要读写 HDFS，即使某些 MapReduce 任务产生的数据仅是临时数据。这就导致了 MapReduce 框架在表达具有依赖关系的作业时是低效的，在算法执行过程中需要读写数据，产生大量的 I/O 开销，算法效率低下。本文提出了一种适用于多个具有依赖关系的 MapReduce任务的计算框架——基于数据流的计算框架。该框架将数据处理过程不再按照单任务建模，而是作为一种数据流图来处理。基于数据流的计算框架把 MapReduce 过程拆分成若干个子过程，同时可以把多个具有依赖关系的MapReduce任务组合成一个较大的 DAG 任务，减少了多个具有依赖关系 MapReduce 任务之间的文件存储。合理组合各个子过程，也可以减少任务的执行时间。

k-means 算法从 $n$ 个数据对象中随机地选取 $k$ 个对象作为初始聚类中心，这就导致了聚集结果的好坏以及算法的效率直接受到初始聚类中心选择的影响。本文采用基于多次采样的方法对随机选取初始聚类中心的方式进行了改进，尽量使得最初的聚类中心与实际数据分布的聚类中心相一致。基于采样确定初始聚类中心的方法不仅能够提高聚类的准确率，而且能够减少算法的迭代次数。当数据量巨大时，迭代次数的减少能够大大降低计算以及 I/O 开销，提高整个算法的效率。

本文第 2 节介绍 k-means 聚类已有的一些相关工作：第 3 节简单介绍了传统 k-means 聚类算法；第 4 节提出基于 MapReduce 的 k-means 算法；第 5 节首先提出基于数据流的计算框架，再基于该框架提出聚类中心选择算法，然后提出 DK-means 算法；第 6 节给出了实验结果并分析。

## 2 相关工作

在过去许多年里，聚类问题是一个数据管理和数据挖掘领域研究者们广泛研究的问题。k-means 是最为经典的基于划分的聚类方法。由于该算法简单高效，在对大规模数据进行聚类时被广泛应用。但 k-means 算法自身存在一些缺陷，例如需要由用户指定 $k$ 值、初始聚类中心随机选择、产生局部最优解甚至无解等。目前，许多算法均围绕着 k-means 算法进行扩展和改进。文献[9]考虑关系数据库管理系统的需求，研究了基于磁盘的 k-means 算法。X-means[10] 是 k-means 的一个变种，有效解决了 $k$ 值需要事先指定的问题，该算法能够预估聚类的个数。Joshua 等人[11] 基于 k-means 提出的聚类算法能够自动计算变量的权重。Alsabti 等人[12] 提出了基于 $k$-$d$ 树数据结构的算法，$k$-$d$ 树对数据空间进行划分，将每一个划分当成一个独立的单元进行处理。这种批处理的方式能够大大减少计算量。王守强等人[13] 提出了近似比为常数的 k-means 算法，其通过不同概率选取初始 $k$ 个点，保证了以一定概率分别属于不同最优聚类簇的 $k$ 个点，以这 $k$ 个点作为初始中心点对输入点集进行交换，分别执行局部搜索。文献[14]提出了基于遗传算法的全局搜索能力来解决初始聚类中心选择的敏感性问题。

国内外研究者提出的以上算法都是基于集中式单节点环境下的。近几年，数据规模呈爆炸式增长，聚类计算任务所面临的数据规模越来越庞大，分布式并行的 k-means 算法越来越受到了人们的关注。文献[15]提出了基于消息传递模型的 k-means 算法，主要研究在消息传递模式下算法的效率和可扩展性的提高。

MapReduce 云计算框架作为当下驾驭大型计算机集群能力的一种流行方式得到了重视。Hadoop 是 MapReduce 计算框架的开源实现版。文献[16]提出了基于 MapReduce 的并行 k-means 算法 PKmeans，但 PKmeans 算法没有考虑初始聚类中心的选取问题，因此效率不高。江小平等人[17] 也做了类似工作，存在同样的问题。文献[18]在 MapReduce 上实现了近似的 k-means 算法，使用 LSH 和剪枝策略极大地提高了算法的效率，但分类结果的正确性并不能得到保证。

## 3 传统的 k-means 聚类算法

k-means 算法是最为经典的基于划分的聚类方法，其基本思想是：选取 $n$ 个数据对象中的 $k$ 个作为初始聚类中心，对于数据集中剩余的每个对象，根据其与各个聚类中心的距离将每个对象划分到最近的类簇中。重新计算每个聚类的均值，更新聚类中心的值。重复这一过程，直到标准准则函数收敛为止。通常采用误差平方和准则函数作为聚类准则函数，误差平方和函数定义为：$V=\sum_{i=1}^{k}\sum_{x\in S_i}(x-\mu_i)^2$，其中 $V$ 是数据集中对象与所在类簇的中心的平方差的总和。$V$ 越大说明类簇内的对象与中心的距离越大，类簇内的相似性越低，反之则说明类簇内的相似性越高。$k$ 是类簇的个数，$x$ 是类簇内的一个对象，$S_i$ 表示第 $i$ 个类簇，$\mu_i$ 是 $S_i$ 类簇的聚类中心。

文献[17]给出了 k-means 算法的过程：

输入：聚类个数 $k$，包含 $n$ 个数据对象的样本集。

输出：满足方差最小标准的 $k$ 个聚类。

处理流程：

(1)从 $n$ 个数据对象中任意选择 $k$ 个对象作为初始聚类的中心；

(2)根据每个聚类中所有对象的均值(聚类中心)，计算样本集中每个对象与这些聚类中心的距离，将每个对象指派给距离最近的类簇；

(3)重新计算每个类簇的均值，更新类簇均值(聚类中心)；

(4)重复步骤(2)和(3)，直到聚类准则函数不再发生变化为止。

串行 k-means 算法的时间复杂度较高，为 $k\times n\times i\times O$。其中 $k$ 为用户指定的参数，即所期望的簇的个数；$n$ 为样本集中数据对象的个数；$i$ 为算法总共迭代的次数；$O$ 为计算样本集中对象与聚类中心距离的时间复杂度。

## 4 基于 MapReduce 的 k-means 聚类算法

MapReduce 云计算框架是当下驾驭大型计算机集群能力的一种流行方式。MapReduce 以数据为中心思想，使得程序员只需要专注于应用到数据记录集上的转换操作，而关于分布式计算、网络通信、容错等细节统统交由 MapReduce 框架进行处理。

### 4.1 编程模型

在 MapReduce 计算框架下，程序员需要将计算表达为一

系列的任务。任务的输入规定了产生 key-value 对的规范。每个任务都包括了两个阶段：Map 阶段和 Reduce 阶段。Map 阶段，使用用户自定义的 map 函数对每条输入记录进行处理，产生一系列的中间结果 key-value 对。Reduce 阶段，map 将具有相同 key 值的中间结果 key-value 对输出给同一个 reduce，使用用户自定义的 reduce 函数对相同 key 值的中间 values 进行处理。MapReduce 能够自动将这些函数执行并行化，并保证容错。

### 4.2 基于 MapReduce 框架的 k-means 算法

在 k-means 算法中，计算样本集中对象与聚类中心之间的距离这个基本操作是最为耗时的部分。由于在样本集中某个对象与 $k$ 个聚类中心进行距离比较的同时，样本集中其他对象也可以与 $k$ 个聚类中心进行距离比较，因此这个基本操作很容易并行化处理。

基于 MapReduce 框架的 k-means 聚类算法思想：一个 MapReduce 任务对应了串行 k-means 算法的一次迭代过程。在 Map 阶段，进行样本集中数据对象与 $k$ 个聚类中心的距离比较。在 Reduce 阶段进行新的聚类中心的计算。图 1 给出了基于 MapReduce 框架的 k-means 聚类算法的处理流程。

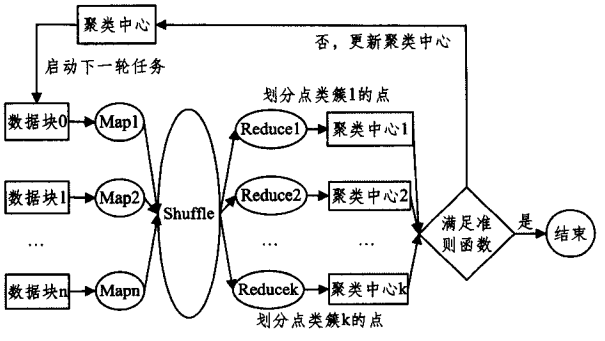


图 1 基于 MapReduce 框架的 k-means 聚类算法

基于 MapReduce 的 k-means 算法的过程：

输入：聚类个数 $k$，包含 $n$ 个数据对象的样本集。

输出：满足方差最小标准的 $k$ 个聚类。

处理流程：

(1)从 $n$ 个数据对象中任意选择 $k$ 个对象作为初始聚类的中心；

(2)在 Map 阶段，计算样本集中每个对象与这些聚类中心的距离，将每个对象指派给距离最近的类簇；

(3)在 Shuffle 阶段，将每个对象 shuffle 到处理对应类簇的 Reduce；

(4)在 Reduce 阶段，重新计算类簇的聚类中心；

(5)计算误差平方和函数，若未收敛，则以新的聚类中心重新聚类，重复(2)到(4)；否则算法结束。

在 Reduce 任务结束之后，根据误差平方和函数：$V=\sum_{i=1}^{k}\sum_{x\in S_i}(x-\mu_i)^2$ 的收敛情况，决定是否启动下一轮 MapReduce 任务：如果 $V$ 收敛，则算法结束，产生输出；如果 $V$ 不收敛，则用 Reduce 阶段计算的聚类中心作为新的聚类中心，启动下一轮 MapReduce 任务，直到 $V$ 收敛结束。

在 MapReduce 计算模型中，将传统 k-means 聚类算法由单个节点完成的任务并行化，交由多个计算节点来协同完成。这样，k-means 聚类算法中最为耗时的计算部分（$k\times n\times i$ 次数据对象与聚类中心的距离计算）将分散由多个计算节点来共同完成。简单假设，每个计算节点完成 $M$ 个 map 任务，则基于 MapReduce 框架的 k-means 聚类算法的时间复杂度将由 $k\times n\times i\times O$ 降为 $k\times n\times i\times O/M$。

## 5 基于数据流的 k-means 聚类算法

本节对基于 MapReduce 框架的 k-means 聚类算法做进一步的优化，优化包括了两个方面。首先，提出了一个新的基于数据流的计算框架。其次，在该框架上提出了初始聚类中心选择算法；最后提出了基于数据流的 k-means 聚类算法 DKmeans。

### 5.1 基于数据流的计算框架及 IPO 运行模型

MapReduce 计算模型将计算过程抽象成 Map 和 Reduce 两个阶段，并通过混洗机制将两个阶段连接起来。但在一些应用场景中，为了套用 MapReduce 模型解决问题，不得不将问题分解成若干个有依赖关系的子问题，每个子问题对应一个 MapReduce 作业。以上运行作业的方式效率较低，根本原因是作业之间的数据不是直接流动的，而是借助 HDFS 作为共享数据存储系统，即一个作业将处理后产生的数据写入 HDFS，另一个依赖于该作业的作业需再从 HDFS 上重新读取数据进行处理。显然更高效的方式是第一个作业直接将产生的数据传输给依赖它的作业，从而可以大大减少 I/O 使用。

基于数据流的计算框架直接源于 MapReduce 框架，将 Map 和 Reduce 两个操作进一步拆分，即 Map 被拆分成 Input、Processor、Sort、Merge 和 Output，Reduce 被拆分成 Input、Shuffle、Sort、Merge、Processor 和 Output 等。这些分解后的元操作可以灵活组合，产生新的操作，这些操作经过组装形成一个大的作业。

基于数据流的计算框架不再按照 Map-Reduce 的运行模型，而是使用 Input-Processor-Output（IPO）的运行时模型。MapReduce 只是一种简单的数据处理模型，它将数据处理过程简化为 Map 和 Reduce 两个阶段，这限制了 MapReduce 的计算表达能力。基于数据流的计算框架不同，它可以包含任意多个数据处理阶段，提供了一种更加灵活高效的编程模型。IPO 运行时模型比 MapReduce 具有更好的可扩展性。

基于数据流的计算框架将数据处理不再按照单任务来建模，而是按照数据流图来处理。图 2 中，(a)是基于 MapReduce 计算框架的作业流程图，(b)是基于数据流计算框架的作业流程图。采用基于数据流的计算框架之后则将作业的依赖关系去除，将有依赖关系的作业转换为一个作业。使用基于数据流的计算框架，对 MapReduce 作业依赖关系进行了裁剪，并将多个小作业合并成一个大作业，不仅减少了计算量，而且也减少了 HDFS 读写次数，从而提升了作业的执行性能。

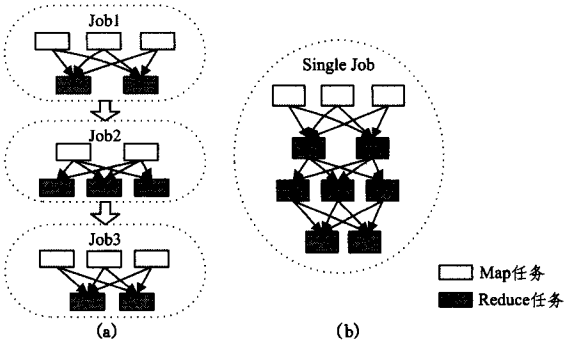


图 2 两种计算框架的作业建模图

### 5.2 基于数据流的初始聚类中心选择算法

原始的 k-means 算法需要从 $n$ 个数据对象中随机地选取 $k$ 个对象作为初始聚类中心。初始聚类中心的选择又直接关系到聚集结果的好坏和算法的效率，因此需要对随机选择初始聚类中心的方式进行改进。本文采用基于采样的方法选取初始聚类中心。

对数据集进行随机采样，使得采样的数据能够保留原始数据集中数据分布的特征。分别采用 k-means 算法对采样数据和原始数据集进行了聚类，结果发现所产生的聚类中心相差不大，因此可以认为通过随机采样方式得到的数据能够反映原始数据集中数据的分布，基于采样的方法适用于选取初始聚类中心。由于采样具有随机性，每次采样的数据不会相同，产生的初始聚类中心也可能不同，而不同的初始聚类中心又会导致算法的效率不稳定，产生的聚类结果也会产生差异。因此，为了减少随机采样所导致的一系列影响，采用了多次取样的方法：从原始数据集中进行 $R$ 次独立随机采样，获取 $R$ 个随机采样样本。对这 $R$ 个样本分别采用 k-means 算法得到样本的聚类中心。根据聚类准则函数，计算 $R$ 组不同聚类中心的误差平方和函数，结果最小的那组聚类中心被选作整个数据集的初始聚类中心。

基于多次采样确定初始聚类中心的算法，是在基于数据流的计算框架上实现的，将多次采样的处理过程并行化。其处理过程如下：

输入：聚类个数 $k$，包含 $n$ 个数据对象的样本集。

输出：$k$ 个初始聚类中心。

对于数据集 $V$ 中每个点 $p$，产生 $R$ 个随机数$\{v_1,\cdots,v_R\}$；

若$\{v_1,\cdots,v_R\}$中的 $v_i$ 大于概率 $q$，则将 $p$ 点放入第 $i$ 个样本中；

产生 $R$ 个采样样本，并混洗到 $R$ 个 Reduce 任务中；

对每个样本进行 k-means 聚类，得到聚类中心；

计算 $R$ 组聚类中心的误差平方和，选择结果最小的一组聚类中心作为初始聚类中心。

### 5.3 基于数据流的 k-means 算法(DKmeans)

由于初始聚类中心的选择直接关系到聚集结果的好坏和算法的效率，因此 DKmeans 算法包括了两个任务：1)初始聚类中心的选取；2)原始数据集的 k-means 聚类。这两个任务都是在基于数据流的计算框架上实现的。其中任务 1 即使用基于数据流的初始聚类中心选择算法选出初始聚类中心；任务 2 使用任务 1 生成的聚类中心作为初始聚类中心，对原始数据集进行 k-means 聚类，该聚类过程由算法 1 和算法 2 实现。

**算法 1** DKmenas 基于聚类中心划分算法

```
输入:数据集及聚类中心
输出:数据集划分
1. min-distance=MAXDISTANCE;
2. foreach point p in V do
3.   for(i=0;i<k;i++) do
4.     if (distance(p,c[i])<min-distance)
5.       min-distance=distance(p,c[i]);
6.   cID=i;
7.   Emit(p,cID);
```

**算法 2** DKmenas 聚类中心重计算算法

```
输入:同一个类簇的所有数据记录
输出:新的聚类中心
1. foreach point p in cluster do
2.   Num++;
3.   for(i=0;i<dimension;i++) do
4.     sum[i]+=p[i];
5. for(i=0;i<dimension;i++) do
6.   newmean[i]=sum[i]/Num;
7. Emit(cID,newmean);
```

算法 1 用于将数据集中的数据记录划分到对应的类簇中，算法 2 用于计算类簇的聚类中心。算法 1 和算法 2 是在基于数据流的计算框架上实现的，与在 MapReduce 计算框架上不同，其不再将一次迭代过程作为一个 MapReduce 任务，而是将整个 k-means 过程看成一个大的计算任务，如图 3 所示，而这个大的计算任务包括了若干小的计算任务，这些小的计算任务就包括数据记录按类簇进行划分和计算类簇的聚类中心。这些小的计算任务之间存在着一定的依赖关系，将这些小的计算任务按照数据流图来建模。

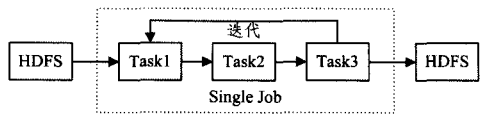


图 3 基于数据流框架的 DKmeans 算法流程

### 5.4 代价分析

改进的 DKmeans 聚类算法在基于采样的初始聚类中心的选取过程中，由于采样的数据样本远远小于原始数据集，因此迭代的次数很少，执行速度很快。当原始数据集的数据量非常大时，基于采样选取初始聚类中心的时间可以忽略不计，在基于数据流的计算模型中，由多个计算节点协同完成 k-means 聚类算法。假定每个计算节点完成 $M$ 个任务，则基于数据流计算框架的 k-means 聚类算法的时间复杂度为 $k\times n\times i\times O/M$。但由于基于采样的初始聚类中心算法所选取的初始聚类中心比随机选取的初始聚类中心更接近真正数据分布的聚类中心，因此迭代次数 $i$ 的值将会大大降低，算法的时间复杂度也会降低，算法的执行效率将会更高。

在基于 MapReduce 计算框架的 k-means 算法中，每一次的迭代过程均需要在 Map 阶段从 HDFS 读取整个数据集，在 Reduce 阶段需要将最终结果写入到 HDFS 中。由于 HDFS 读写是基于磁盘的，因此读写速度相对较慢，而且在 HDFS 中写入数据都是默认写 3 份的。每次迭代产生的结果只是中间结果，并不需要持久保存，因此基于 MapReduce 计算框架的 k-means 算法的 I/O 开销较大，输入代价为 $O(i*n)$，输出代价为 $O(3i*n)$。而基于数据流计算框架的 k-means 算法是按照数据流来建模的，各个任务之间的数据是直接流动的，即无需借助 HDFS 作为共享数据存储系统，一个作业将处理后产生的数据直接传递给下一个作业。DKmeans 算法任务 1 和任务 2 的输入代价均为 $O(n)$，总输入代价为 $O(2n)$，任务 1 的输出代价忽略不计，任务 2 的输出代价为 $O(3n)$。在数据量特别大的情况下，I/O 已成为性能瓶颈，DKmeans 算法 I/O 代价与基于 MapReduce 计算框架的 k-means 算法相比明显减小，从而提高了整个算法的执行效率。

## 6 实验结果与分析

### 6.1 实验设置

集群有 8 个节点；CPU：Xeon E5-2620(双核 2.00GHz)；

内存:8GB;硬盘:6TB(硬盘实际可用空间为 44.03TB);操作系统:CentOS6.4(64bit);分布式文件系统:HDFS;MapReduce 框架开源实现版:Hadoop;基于数据流的计算框架基于 MapReduce 框架,其中一个节点为 master,剩下的节点均为 worker 节点。

实验数据使用移动用户数据,对移动用户数据进行聚类得到不同特征类型的用户群组。每条数据包括了 35 个数值类型的属性。为了测试性能并与其他算法进行性能对比,从数据集中随机提取了 3 组数据集,如表 1 所列,要求生成 5 个聚类类别。

表 1 实验数据集

| 数据集 | 文件大小/MB | 记录条/$10^6$ | 数据块数 |
|---|---|---|---|
| A | 2023.46 | 5.321423 | 78 |
| B | 4101.34 | 10.824012 | 164 |
| C | 8221.01 | 20.912323 | 322 |

### 6.2 结果分析

(1)可扩展性实验结果与分析

图 4 示出了在节点个数分别为 2,4,8 时,算法的执行时间。从图中可以看出,在不同数据规模下,随着集群中节点数目的增长,DKmeans 算法执行时间呈近线性的递减。增加节点能够有效提高系统对同数据规模的处理能力。

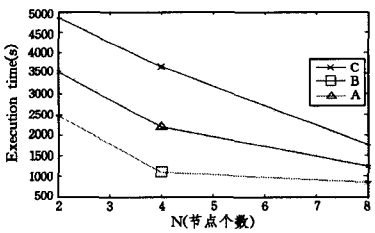


图 4 不同数据规模下时间性能随节点数目的变化

图 5 示出了在计算节点固定为 8、数据量不同时,算法的执行时间。从图中可以看出,随着数据规模的不断增大,DKmeans 算法执行时间随着数据规模增长呈近线性的增长,这说明该 DKmeans 算法具有较好的可扩展性。计算节点是衡量可扩展性的一个重要维度,若计算任务不变,当计算节点增加一倍时,则执行时间应当减少一半。然而在实际算法运行过程中,受容错、数据通信等各方面因素的影响,其执行时间以一种近线性的速率递减。当数据量较小时,如图 5 中数据量为 A 时,随着节点的增加,会出现资源闲置的情况,效率并不能得到进一步的提升。

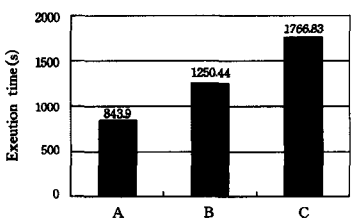


图 5 节点数目为 8 时时间性能随数据规模的变化

(2)加速比性能实验结果与分析

最完美的并行算法应该能得到线性加速比,即当系统中节点数量增加 $k$ 倍时,加速比就为 $k$。但由于随着节点的增加,节点之间的通信代价也会增大,因此通常并行算法很难达到线性加速比。

图 6 示出了在节点个数分别为 2,4,8、数据集分别为 A,B,C 时的加速比。从图中可以看出 DKmeans 算法具有良好的加速比性能(近线性);而且,随着数据集的增大,加速比性能更优。因此,DKmeans 算法可用于处理大规模数据集。当数据量较小时,如图中数据量为 A 时,随着节点的增加,会出现资源闲置的情况,因此加速比不会随着节点增加而得到进一步的提升。

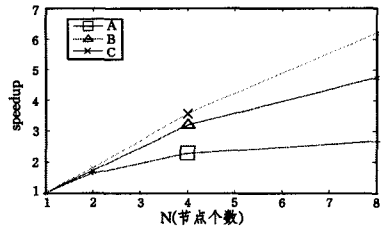


图 6 DKmeans 算法的加速比

(3)与 PKmeans 算法的性能比较

图 7 示出了在节点数目分别为 2,4,8 时,不同数据规模下,文献[16]提出的 PKmeans 算法(图中表示为 MR k-means)与 DKmeans 算法的执行时间。由于在 MapReduce 框架中,每个 MapReduce 任务都需要读写 HDFS,即使某些 MapReduce 任务产生的数据仅是临时数据。显然,这种表达作业依赖关系的方式效率不高。而且 PKmeans 算法的初始聚类中心是随机选取的,因此效率更低。从图中可以看出,在处理相同数据规模的数据时,DKmeans 算法效率比 PKmeans 算法效率明显更高;而且随着数据量的增大,基于数据流的计算框架比基于 MapReduce 框架的优势将会更加明显,这是由于基于数据流的计算框架减少了任务间数据的读写,因此大大降低了 I/O 开销。

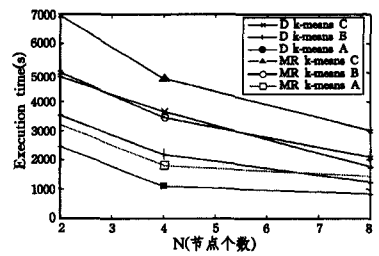


图 7 与 PKmeans 算法的性能比较

(4)与随机选择初始聚类中心算法的性能比较

图 8 示出了在节点数目分别为 2,4,8 时,不同数据规模下,DKmeans 算法分别采用随机选择初始聚类中心(R k-means)和基于采样选择初始聚类中心(D k-means)两种方法的执行时间。在选择初始聚类中心时,随机选择要比基于采样选择更节省时间。但是由于初始聚类中心的选择又直接关系到聚集结果的好坏和算法的效率,因此选择好的初始聚类中心是有助于提高算法效率的。从图中可以看出,采用基于采样选择初始聚类中心的算法的性能始终是优于采用随机选择算法的;而且,当数据规模越大,基于采样选择初始聚类中心算法的优势更加明显,因为好的初始聚类中心能够大大减少算法的迭代次数,而随着数据规模的增大,一次迭代的代价会更大。

(下转第 265 页)

45(4):1036-1040

[14] Sternberg R J. Innovation: Lighting the creative spark[J]. Nature, 2010, 468(7321): 170-171

[15] Liang J J, Qu B Y, Suganthan P N, et al. Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization [R]. Zhengzhou: Computational Intelligence Laboratory, Zhengzhou University, Singapore: Nanyang Technological University, 2013

[16] Caraffini F, Neri F, Cheng J, et al. Super-fit Multicriteria Adaptive Differential Evolution[C]// 2013 IEEE Congress on Evolutionary Computation(CEC). IEEE, 2013: 1678-1685

[17] Tanabe R, Fukunaga A. Evaluating the performance of SHADE on CEC 2013 benchmark problems[C]// 2013 IEEE Congress on Evolutionary Computation(CEC). IEEE, 2013: 1952-1959

[18] El-Abd M. Testing a particle swarm optimization and artificial bee colony hybrid algorithm on the CEC13 benchmarks[C]// 2013 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2013: 2215-2220

(上接第 239 页)

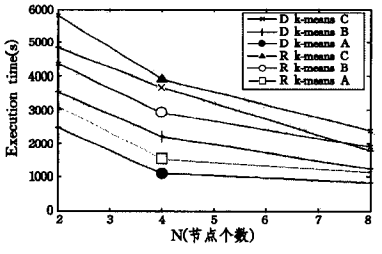


图 8 与随机选择初始聚类中心算法的性能比较

**结束语** 本文研究分布式环境下大规模数据集的 k-means 聚类问题，首先提出了当下流行的 MapReduce 计算框架下的 k-means 算法。MapReduce 框架下由于每个MapReduce 任务都需要读写分布式文件系统，因此 MapReduce 表达作业之间依赖关系时是低效的，这就导致现有的基于MapReduce 的算法的效率不高。本文在 MapReduce 框架的基础上提出了基于数据流的框架，并在该框架上提出了一种基于数据流计算框架的 k-means 算法 DKmeans，该算法将k-means 聚类过程归结为两个步骤，从而可以减少读写分布式文件系统的次数，有效降低查询过程中的 I/O 代价。针对步骤一中初始聚类中心选择的问题，本文提出了基于采样的初始聚类中心选择算法。随着摩尔定律渐渐失效以及待处理数据的爆炸式增长，基于数据流计算框架的 k-means 算法能高效地处理大规模数据的 k-means 聚类问题。实验表明，本文算法具有良好的可扩展性和执行效率，具有实用价值。

## 参 考 文 献

[1] Han Jia-wei, Kamber M. Data mining concepts and techniques, second edition[M]. Elsevier (Singapore) Pte Ltd, 2006: 251-263

[2] Kriegel H P, Kröger P, Zimek A. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering[J]. ACM Transactions on Knowledge Discovery from Data (TKDD), 2009, 3(1): 1

[3] Forgy E. Cluster analysis of multivariate data: Efficiency vs. Interpretability of classifications [J]. Biometrics, 1965, 21(3): 768

[4] Malewicz G, Austern M H, Bik A J C, et al. Pregel: a system for large-scale graph processing [C] // Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data. ACM, 2010: 135-146

[5] Wang J, Su X. An improved K-Means clustering algorithm[C]// 2011 IEEE 3rd International Conference on Communication Software and Networks (ICCSN). IEEE, 2011: 44-46

[6] Kumar N S, Rao K N, Govardhan A, et al. Undersampled K-means approach for handling imbalanced distributed data[J]. Progress in Artificial Intelligence, 2014, 3(1): 1-10

[7] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1): 107-113

[8] Apache. Hadoop[EB/OL]. (2014-4-10)[2014-4-22]. http://hadoop.apache.org/

[9] Ordonez C, Omiecinski E. Efficient disk-based K-means clustering for relational databases[J]. IEEE Transactions on Knowledge and Data Engineering, 2004, 16(8): 909-921

[10] Pelleg D, Moore A W. X-means: Extending K-means with Efficient Estimation of the Number of Clusters[C]// ICML. 2000: 727-734

[11] Huang J Z, Ng M K, Rong H, et al. Automated variable weighting in k-means type clustering[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2005, 27(5): 657-668

[12] AlSabti K, Ranka S, Singh V. An efficient space-partitioning based algorithm for the K-means clustering[M]// Methodologies for Knowledge Discovery and Data Mining. Springer Berlin Heidelberg, 1999: 355-360

[13] 王守强，朱大铭，韩爱丽. 基于初始点选取的 k-means 聚类近似常数算法[J]. 计算机研究与发展，2007(z2)：69-74
Wang Shou-qiang, Zhu Da-ming, Han Ai-li. A Constant Approximate Alogritnm for K-Means Problem Based on Inital Point Selecting [J]. Journal of Computer Research and Development, 2007(z2): 69-74

[14] 李飞，薛彬，黄亚楼. 初始中心优化的 K-Means 聚类算法[J]. 计算机科学，2002，29(7)：94-96
Li Fei, Xue Bin, Huang Ya-lou. K-Means Clustering Algorithm with Refined Inital Center[J]. Computer Science, 2002, 29(7): 94-96

[15] Dhillon I S, Modha D S. A data-clustering algorithm on distributed memory multiprocessors [M] // Large-Scale Parallel Data Mining. Springer Berlin Heidelberg, 2000: 245-260

[16] Zhao W, Ma H, He Q. Parallel k-means clustering based on mapreduce[M] // Cloud Computing. Springer Berlin Heidelberg, 2009: 674-679

[17] 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
Jiang Xiao-ping, Li Cheng-hua, Xiang Wen, et al. Parallel implementing k-means clustering algorithm using MapReduce programming mode [J]. Journal of Huazhong University of Science and Technology (Natural Science Edition), 2011, 39(1): 120-124

[18] Li Q, Wang P, Wang W, et al. An Efficient K-means Clustering Algorithm on MapReduce [C]// Database Systems for Advanced Applications. Springer International Publishing, 2014: 357-371