基于 MapReduce 的 K_means 并行算法及改进①

衣治安, 王 月

(东北石油大学 计算机与信息技术学院, 大庆 163318)

摘 要: 针对传统 k_means 聚类算法在处理海量数据时所面临的内存不足、运算速度慢等问题, 提出了一种基于 MapReduce 的 K_means 并行算法, 同时为了改善 k_means 算法在初始值确定方面的盲目性, 采用 canopy 算法进行改进. 实验结果表明, 基于 MapReduce 的 K_means 并行算法和改进后的算法均能产生良好的聚类效果, 不仅提高了聚类质量, 而且在处理大数据集方面, 改进后的算法的还能够得到趋近于线性的加速比.

关键词: MapReduce; k-means 算法; canopy 算法; 并行计算; 聚类

Parallel K-Means Algorithm and Improved Based on MapReduce

YI Zhi-An, WANG Yue

(Northeast Petroleum University, College of Computer and Information Technology, Daqing 163318, China)

Abstract: In view of the problems that traditional k-means clustering algorithm faces in dealing with mass data, such as running out of memory, the operating in slow speed and so on, this paper proposes a parallel k-means algorithm based on MapReduce. At the same time, in order to overcome the blindness of the k-means algorithm in terms of determining the initial value, we use the canopy algorithm to improve the insufficient. The experimental results show that the parallel k-means algorithm based on MapReduce has an effect on clustering before and after the improvement, not only the quality of the clustering has been increased, but in terms of processing large datasets. The speed-up ratio of the improved algorithm can get closer to the linear.

Key words: MapReduce; k-means algorithm; canopy algorithm; parallel computation; cluster

随着社会信息化的不断发展, 人们日常生活中所产生的信息也越来越多, 网络上的数据正以成倍的速度日益增长, 如何从海量数据中检索到有价值的信息已成为人们最迫切的需求. 聚类分析作为数据挖掘的重要组成部分, 已经被各个领域广泛使用并取得了一定成果, 但随着数据规模的高速增长, 其弊端也日益显露出来, 一是聚类速度和效率的问题; 二是受制于计算机内存条件的限制. 为了解决传统串行聚类算法的弊端, 本文采用并行计算的思想, 基于目前非常流行的 Apache 提出的 Hadoop[1]分布式框架, 研究了 MapReduce 模型下 K_means 算法的并行实现, 同时完成了基于 MapReduce 模型的改进算法的研究.

1 MapReduce编程模型

MapReduce 是 Google 提出的一种用于并行处理大数据集的分布式计算框架, 它主要通过 Map(映射)和 Reduce(化简)两个步骤实现对大规模数据的并行处理[2,3]. 简单来说, Map 是一个切分和解析的过程, 负责将一个大的数据集切分成多个小的数据块, 并映射到若干节点上进行并行处理, Reduce 是一个归纳和聚合的过程, 负责将 Map 阶段操作的结果利用多个并行的 reduce 化简函数进行汇总, 同时在整个 MapReduce 的执行过程中都以键值对(key/value)作为输入和输出的. 图 1 描述了 MapReduce 的运行过程.

在图中可以看出, 在将数据分发到 Map 之前会有一个切分(partion)的过程, 这个过程将原始的数据文件切分成了 N 个数据块 splits, 在经由 JobTracker 指定执行 Map 任务的 TaskTracker 读取后被转换成了 key/value 键值对, 每个 map 之间都是独立、并行执行的, 它会接收这些键值对作为输入数据, 并将得到的

①收稿时间:2014-10-11;收到修改稿时间:2014-11-13

中间结果(以 key/value 键值对的形式输出)暂时写入本地内存的缓冲区中, 这些缓冲区的内容会被分区函数划分到不同的分区中, 最后写入本地磁盘, 随后执行 reduce 任务的节点会远程读取保存到本地磁盘的键值对, 同时按照 Key 值进行排序并分组, 每个 reduce 负责对一个组别进行运算处理, 然后将运算结果输出到最终文件中, 在所有的 map 和 reduce 任务都执行完成以后, JobTracker 会告诉应用程序进行下一阶段程序的执行.

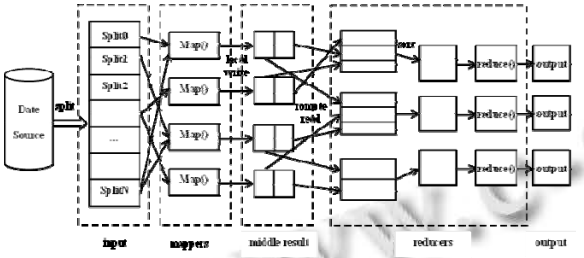

图 1 MapReduce 执行流程图

2 K_means聚类算法

K_means 算法是传统聚类分析方法中划分聚类法的一种[4], 因为算法的思想相对比较简单、对数据输入顺序不敏感、计算效率比较高等特点而被广泛使用. K_means(亦称 K 均值)算法的基本思路是: 对于一个给定的数据集 S, 首先接受参数 K 作为初始类簇的数目, 并将随机选取的 K 个数据对象作为 K 个类簇的初始聚类中心, 对于数据集 S 中剩下的数据分别计算它们到这 K 个聚类中心的距离, 然后分配到距离它们最近的聚类中心所在的类簇中, 重新计算 K 个类簇中所有数据的平均值, 结果作为新的聚类中心, 重复这个过程直到聚类中心不再发生变化. 该算法的伪代码如下[5]:

输入: 数据集 S, 簇个数 k

输出: k 个聚类

过程:

(1)从 S 中任选 k 个数据作为初始中心

(2)do{

(3)foreach (s in S)

(4)foreach(center in center_list)

(5)计算中心点以外的数据到每次所迭代产生 new_center 的距离(第一次为初始 center)

(6)将数据 s 划分到和 s 距离最近的 center 的类簇 datalist 中

(7)计算每个类簇的平均值, 并把它赋值给新聚类中心

(8)}while(聚类中心不再发生变化)

(9)输出 new_center_list 和 datalist

k_means 算法的时间复杂度相对比较高, 为数据集 S 的总个数 N×聚类个数 K×算法的迭代次数 T, 如果数据是多维的, 算法的复杂度 O 还需要乘以数据维度 M[6]. 可以看出该算法的执行还是比较耗时的, 由此想到如果可以并行的完成数据与中心点的距离计算, 无疑可以很大程度上的提高该算法的执行效率.

3 基于MapReduce模型的K_means聚类算法

3.1 基于 MapReduce 的 K_means 算法思想

K_means 算法并行化的思想: 对算法每迭代一次就会对应启动 MapReduce 计算过程一次, 完成待处理的数据记录到数据中心距离的计算, 进而将数据重新分配并划分到不同的聚类中, 达到进一步更新聚类中心的目的. 在 MapReduce 编程模型下, 首先需要对需要聚类的数据集进行一些预处理操作, 转化为可以被 MapReduce 读取的形式, 即将数据记录进行按行切片, 对于每个片间的数据彼此是独立无关的, 整个过程无需人工编写代码, MapReduce 的运行环境会自动为我们完成, 从而该算法的并行化可以主要依靠 Map 函数和 Reduce 函数来实现.

(1) Map 函数

Map 函数需要完成的主要任务就是计算数据集中所有数据对象到每个聚类中心距离, 同时将数据对象重新划分聚类簇, 它接收的输入是< key, value >键值对集合, 其中 key 指的是上一轮迭代(第一次指初始聚类)的聚类中心, value 指的是待聚类的数据对象块, map 函数对输入的每条记录会进行三部分处理, 首先读取聚类簇的中心点集信息, 计算每条输入记录到各个中心点之间的距离并标记类别, 然后把标记好的记录重新分配给距离它最近的簇类, 最后输出一系列键值对, 形式为<聚类簇 ID, 数据对象属性信息>, 并将这些键值对交给对应的 Reduce 做下一步处理.

(2) Reduce 函数

Reduce 函数的主要任务就是以 Map 阶段处理得到的键值对为基础, 计算出新的聚类中心点, 然后将这些新得到的中心点作为输入数据提交给下一轮的

MapReduce 使用. reduce 函数与 map 函数一样接收的也是键值对形式的输入, 输入形式为<聚类簇 ID, {数据对象集}>, 这里每个 reduce 任务处理的都是具有相同 key 值的记录(即具有相同类簇 ID 的记录), 同时还会累加 key 值相同的记录数并对增加的数据对象求和, 然后针对每个簇, 计算簇中所有数据对象的平均值, 同时将这个平均值作为参照点, 找到距离这个参照点最近的数据对象作为新聚类簇的中心点, 然后输出形为<聚类簇 ID, 中心点>的<key, value>键值对, 最后本次得到的聚类中心与上一轮 MapReduce 任务得到的聚类中心比较, 如果满足收敛性(即变化小于给定的阀值), 则聚类结束, 否则将本次得到的聚类中心作为新一轮 MapReduce 任务的输入, 重复聚类直到满足收敛性. 在 Reduce 函数的执行过程中, 可以对 Map 任务得到的中间节点进行分组或排序处理以提高执行效率.

3.2 基于 MapReduce 的 K_means 算法流程

基于 MapReduce 的 K_means 算法实现并行化分为两个阶段, 第一阶段是对原始数据的处理, 第二部分是 K_means 并行算法的实现, 其流程如图 2 所示.

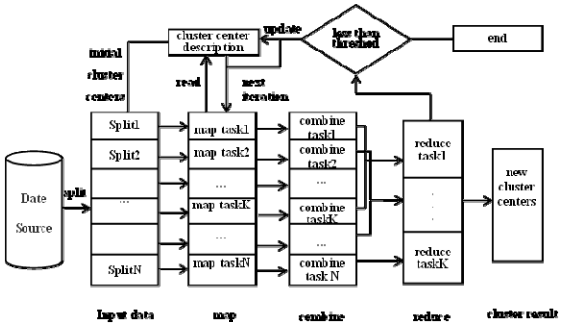

图 2　基于 MapReduce 的 K_means 算法并行流程图

第一个阶段 MapReduce 作业主要完成对数据集的切分处理, 并将切分后的分片以<key, value>键值对的形式复制到各个节点. 首先由 JobClient 对数据源进行切片, 切片成功后, JobClient 会将切片信息发送给 JobTracker, 然后在使用时会加载 InputSplit 记录中的切片信息分配给 map 并作为 map 的输入.

第二阶段主要启动 Map 和 Reduce 进行运用算法进行并行化计算, 直到产生高质量的聚类结果. 其中对于每次计算都需要启动多个 Map 和 Reduce 任务, map 任务负责计算读取到的数据块文件中各个数据到达簇类中心点的距离, 该过程采用欧式距离进行相似度的度量, 然后将这些数据对象分配到距离最近的聚类中; reduce 任务负责汇集各个聚类中的数据对象, 计算出新的聚类中心作为输出数据, 同时也作为下一次迭代时的输入数据, 直到聚类中心不再发生变化为止. 这里增加了多个 Combine 任务, 它主要负责对 map 的输出结果做一些处理, 比如合并具有相同 key 值得键值对, 计算各簇类的平均值等等, 从而减轻节点间数据传输的通信负担.

3.3 基于 MapReduce 的 K_means 算法改进

k_means 算法在开始时由于需要在数据集中随机选择 k 个数据对象做为 k 个簇的初始中心, 而 k 值一般是随机选取的[7], 具有一定的盲目性, 同时迭代过程需要的计算量非常大. 所以本文在基于 MapReduce 的 k_means 算法基础上, 考虑使用 canopy 算法对 k_means 算法进行优化, 主要在质心的选取方面进行优化. 采用 canopy 算法首先将数据划分到若干子集(即 canopy)中, canopy 之间可以有交叉, 保证了每个数据对象至少属于一个 canopy, 然后对分布在同一个 canopy 的数据重新计算中心点, 最后根据数据与新中心点间的距离重新划分数据的所属子集, 重复计算中心点和划分数据区域这个过程, 直到中心点不再发生变化为止.

改进后的并行算法流程图如图 3 所示, 其中阶段 I 属于数据预处理, 阶段 II 是 canopy 算法处理, 阶段 III 属于 k_means 实现, 中间数据 S1 是数据集切分后的<key, value>键值对, 中间数据 S2 是经由 canopy 处理后输出的<key, value>键值对, 三个阶段都包括多个 map 和 reduce 任务.

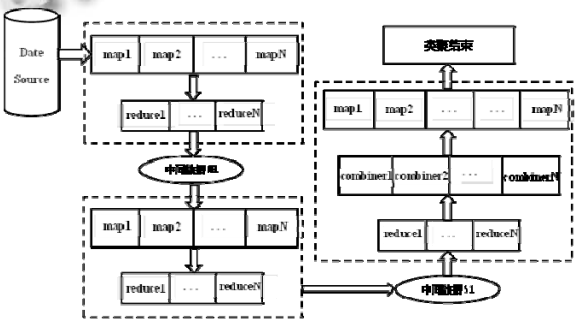

图 3　基于 MapReduce 的 K_means 改进算法流程图

canopy 算法处理的具体实现过程是: (1)canopies 划分, 随机选取一个数据点作为第一个子集的聚类标识 canopyID, 然后 map 需要对数据对象进行判断, 看是否属于之前产生的 canopies, 如果属于其中某一个

canopy 则标识自己的所属关系, 否则要产生一个新的 canopy, 并将本身的数据值作为这个新产生 canopy 的 canopyID, 循环直至所有数据都被划分到相应 canopies 列表中, 其中一个数据可能同时属于多个子集. (2)数据分配, map 会加载 canopies 并判断每个数据的所属关系, 然后输出形式为<数据值, 所属 canopies 的 canopyID>的键值对. (3)结果输出, 将划分所产生 canopies 代替初始的聚类簇, 因为所有数据点已经被划分到相应的 canopies 中, 所以在计算到中心点距离的时候只需计算和该数据同属一个 canopy 的中心点, 然后将新产生的 k_centers 与上一次的 k_centers 比较, 当变化小于给定阀值时迭代结束, 最后 reduce 输出形式为<k_centers, datavalue_list>的键值对. 经过这一步骤不仅提高了运算效率, 同时也强化了 k_means 算法在孤立点处理能力方面的不足.

4 实验与结果分析

本试验是基于 Hadoop 分布式平台完成的[8,9], 在 UCI 上选用了 IRIS(数据集 S1)和 Brest Cancer(数据集 S2)两组数据集对基于 MapReduce 的算法进行测试, S1 包含 150 个样本, 每个样本有 4 个属性, S2 包含 286 个样本, 每个样本有 9 个属性, 本实验分别对并行的 k_means 算法和基于 canopy 的并行 k_means 算法进行测试, 并从加速比和可扩展性两方面对实验结果进行分析.

为了方便对比两个算法的性能, 根据数据聚类过程中迭代的误差和绘制出如图 4 所示的收敛曲线, 由此可以直观的发现经过 canopy 预处理后的聚类结果要比单独的 k_means 并行算法好, 而且收敛的更快.

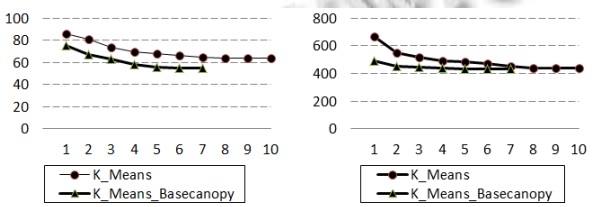

图 4 S1 和 S2 上并行算法收敛曲线图

(1) 算法加速比分析

加速比是验证并行算法的一个重要指标[10], 加速比的计算公式是: TS=T1/TN, T1 指单个节点求解运行的时间, TN 值 N 个节点并行运算需要的时间, 实验从 S1 数据集中随机选出 3 个样本 D1、D2 和 D3, 个数分别为 5000, 10000 和 20000, 经过测试得到如图 5 所示的 k_means 算法改进前后的加速比曲线, 由图中可以看出, 基于 MapReduce 的 k_means 并行算法和改进后的并行算法均有比较良好的加速比, 而且随着节点数目的增加, 加速比的增长速率会逐渐放慢, 主要是因为节点之间的通信量及时间花费逐渐变大, 同时对于数据量较大的样本, 可以看出两个算法执行效率都有明显的提升, 数据规模越大, 其加速比越接近线性增加.

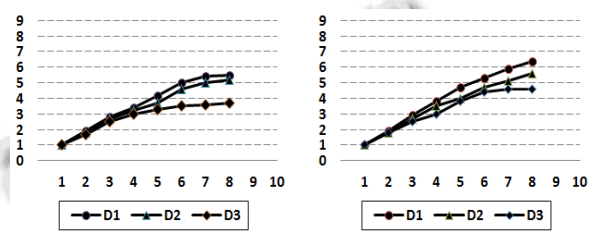

图 5 k-means 算法改进前后加速比曲线图

(2) 算法可扩展性分析

一般情况下, 当运算的节点数目不断增加时, 并行算法的加速比是不能随之无限增大的, 也就是说加速比不能反映集群的利用率情况, 所以本文采用扩展率来反映集群的利用情况, 公式为: $E=T/P$, T 表示算法的加速比, P 表示集群中的节点数目, 从图 6 中可以看出, 第一, 对每个数据集而言, 随着节点数的增加, 并行算法的扩展率曲线逐渐下降; 第二, 数据集的规模越大, 扩展率曲线越趋于平稳; 第三, 数据集规模越小, 当节点增加到某一个数值时, 下降越快, 本实验中可以看出当节点数目为 2 时, D1 样本数据下降的最快, 综上所述, 可以看出改进前后的并行算法都具有良好的可扩展性.

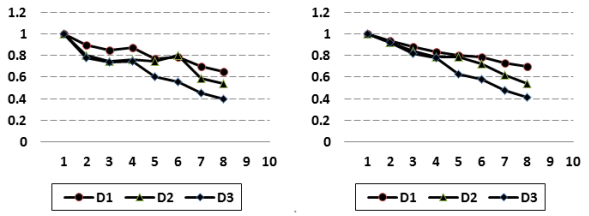

图 6 k-means 算法改进前后扩展率曲线图

参考文献

1 杨宸铸.基于 HADOOP 的数据挖掘研究[硕士学位论文].重庆:重庆大学,2010.

2 Huang WQ, Chen M. Note on: An improved algorithm for the packing of unequal circles within a larger containing

circle. Computers & Industrial Engineering, 2006, 50(3): 338–344.

3 Dean J, Ghemawat S. MapReduce: Simplified Data processing on Large Clusters. Communications of the ACM, 2008, 51(1): 107–113.

4 周锋,李旭伟.一种改进的 MapReduce 并行编程模型.科协论坛,2009(2):65–66.

5 Wegener D. Mock M. Adranale D. et al. Toolkit based high-performance data mining of large data on MapReduce clusters. IEEE International Conference on Data Mining ICDM. Washington. IEEE. 2009. 296–301.

6 吴夙慧,成颖,郑彦宁.K-means算法研究综述.知识组织与知识管理,2011(4):28–34.

7 郑启龙,房明,汪胜.基于 MapReduce 模型的并行科学计算.微电子学与计算机,2009,26(8):13–17.

8 Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. Operating System Design and Implementation, 2004: 137–149.

9 李成华,张新访,金海.MapReduce:新型的分布式并行计算编程模型.计算机工程与科学,2011,33(3):129–135.

10 Kruijf MD, Sankaralingam K. MapReduce for the cell broadband engine architecture. IBM Journal of Research and Development, 2009, 53(5): 747–758.