基于云计算平台 Hadoop 的并行 k-means 聚类算法设计研究

赵卫中[1,4] 马慧芳[2,4] 傅燕翔[3] 史忠植[4]
(湘潭大学信息工程学院 湘潭 411105)[1] (西北师范大学数学与信息科学学院 兰州 730070)[2]
(湘潭大学机械工程学院 湘潭 411105)[3]
(中国科学院计算技术研究所智能信息处理重点实验室 北京 100190)[4]

摘 要 随着数据库技术的发展和 Internet 的迅速普及,实际应用中需要处理的数据量急剧地增长,致聚类研究面临许多新的问题和挑战,如海量数据和新的计算环境等。深入研究了基于云计算平台 Hadoop 的并行 k-means 聚类算法,给出了算法设计的方法和策略。在多个不同大小数据集上的实验表明,设计的并行聚类算法具有优良的加速比、扩展率和数据伸缩率等性能,适合用于海量数据的分析和挖掘。

关键词 云计算,Hadoop 平台,并行 k-means,MapReduce

Research on Parallel k-means Algorithm Design Based on Hadoop Platform

ZHAO Wei-zhong[1,4] MA Hui-fang[2,4] FU Yan-xiang[3] SHI Zhong-zhi[4]
(College of Information Engineering, Xiangtan University, Xiangtan 411105, China)[1]
(College of Mathematics and Information, Northwest Normal University, Lanzhou 730070, China)[2]
(College of Mechanical Engineering, Xiangtan University, Xiangtan 411105, China)[3]
(Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China)[4]

Abstract In the past decades, data clustering has been studied extensively and a mass of methods and theories have been achieved. However, with the development of database and popularity of Internet, a lot of new challenges such as massive data and new computing environment lie in the research on data clustering. We conducted a deep research on parallel k-means algorithm based on Hadoop, which is a new cloud computing platform. We showed how to design parallel k-means algorithms on Hadoop. Experiments on different size of datasets demonstrate that our proposed algorithm shows good performance on speedup, scaleup and sizeup. Thus it fits to data clustering on huge datasets.

Keywords Cloud computing, Hadoop, Parallel k-means, MapReduce

1 引言

聚类是数据挖掘中重要的研究课题之一。所谓聚类,就是将物理或抽象对象的集合组成为由类似的对象组成的多个类或簇的过程。由聚类生成的簇是一组数据对象的集合,同一簇中的对象尽可能相似,而不同簇中的对象尽可能相异[1]。随着数据库技术的成熟和数据应用的普及,商业、企业、科研机构或者政府部门都积累了大量的、以不同形式存储的数据。如何存储、处理这些海量数据,以及进一步从中挖掘出有用的、可以指导应用的知识,成为一个棘手的问题。在面对海量数据时,现有的聚类算法在时间复杂性和空间复杂性上遇到了瓶颈,这也是聚类算法研究领域中亟需解决的问题之一。解决该问题的一个思路就是将并行处理技术应用到聚类中,设计出高效的并行聚类算法,来提高聚类算法处理海量数据时的性能。

云计算作为一种新兴的商业计算模型得到了人们的广泛关注[2-5]。Hadoop 是一个可以更容易开发和并行处理大规模数据的云计算平台,它的主要特点包括扩容能力强、成本低、效率高以及可靠性好等。Hadoop 平台由两部分组成:Hadoop 分布式文件系统(HDFS)[6]和 MapReduce 计算模型[7]。

HDFS 采用 M/S 架构,一个 HDFS 集群是由一个管理节点(Namenode)和一定数目的数据节点(Datanode)组成,每个节点均是一台普通 PC。在使用上,HDFS 与单机上的文件系统非常类似,同样可以建目录,创建、复制、删除文件,查看文件内容等。但其底层实现上是把文件切割成块,然后这些块分散地存储于不同的数据节点上。每个块还可以复制若干

到稿日期:2010-11-01 返修日期:2011-03-21 本文受国家自然科学基金(60933004,60975039,61072085),国家 973 项目(2007CB311004),西北师范大学青年教师科研能力提升计划骨干项目(NWNU-LKQN-10-1),湘潭大学博士启动基金(10QDZ42),湖南省教育厅一般项目(09C967)资助。

赵卫中(1981—),男,博士,讲师,主要研究领域为机器学习、数据挖掘、算法分析与设计,E-mail:zhaoweizhong@gmail.com;马慧芳(1981—),女,博士,副教授,主要研究领域为机器学习、数据挖掘;傅燕翔(1979—),女,讲师,主要研究领域为人机界面交互;史忠植(1941—),男,研究员,博士生导师,主要研究领域为人工智能、机器学习、神经计算、认知科学。

份，存储于不同的数据节点上，以达到容错之目的。管理节点是一个中心服务器，负责管理文件系统的名字空间(Namespace)以及客户端对文件的访问。集群中的数据节点负责管理它所在节点上的存储。管理节点是整个 HDFS 的核心，它通过维护一组数据结构，记录每一个文件被切割成了多少块、这些块可以从哪些数据节点中获得、各个数据节点的状态等重要信息。

MapReduce 是一种高效的分布式编程模型，也是一种用于处理和生成大规模数据集的实现方式。MapReduce 计算模型各个阶段的工作流程如下：

1)Input：一个基于 Hadoop 平台 MapReduce 框架的应用通常需要一对通过实现合适的接口或抽象类提供的 Map 和 Reduce 函数，还应该指明输入、输出的位置和其他一些运行参数。此阶段还会把输入目录下的大数据文件划分为若干独立的数据块。

2)Map：MapReduce 框架把应用的输入看作一组〈*key*，*value*〉键值对。在 Map 这个阶段，框架会调用用户自定义的 Map 函数，处理每个〈*key*，*value*〉键值对。同时生成一批新的中间〈*key*，*value*〉键值对。这两组键值对的类型可能不同。

3)Shuffle：为了保证 Reduce 的输入是 Map 排好序的输出，在 Shuffle 阶段，框架通过 HTTP 为每个 Reduce 获得所有 Map 输出中与之相关的〈*key*，*value*〉键值对；MapReduce 框架按照 *key* 值对 Reduce 阶段的输入进行分组(因为不同 Map 的输出中可能会有相同的 *key*)。

4)Reduce：此阶段会遍历中间数据，对每一个唯一的 key，执行用户自定义的 Reduce 函数。输入参数是〈*key*，{list of values}〉，输出是新的〈*key*，*value*〉键值对。

5)Output：此阶段会把 Reduce 输出的结果写入到输出目录指定的位置。这样，一个典型的 MapReduce 过程就完成了。

2 基于 Hadoop 的并行 *k*-means 聚类算法设计

由上一部分的介绍可以看出，基于 Hadoop 的并行算法设计，用户最主要的工作是设计和实现 Map 和 Reduce 函数，包括输入和输出〈*key*，*value*〉键值对的类型以及 Map 和 Reduce 函数的具体逻辑等。

串行的 *k*-means 算法的步骤为：

1)任意选择 *k* 个样本作为聚簇初始的中心点；

2)迭代；

a)根据每个聚簇的中心点坐标，将每个样本分配给距离其最近的聚簇；

b)更新聚簇的中心点坐标，即计算每个聚簇中所有样本的均值；

3)直到收敛。

从 *k*-means 算法中可以看出，算法中主要的计算工作是将每个样本分配给距离其最近的聚簇，并且分配不同样本的操作之间是相互独立的，因此考虑将这一步骤并行地执行。在每次迭代中，算法执行相同的操作，并行 *k*-means 算法(Parallel KMeans：PKMeans)在每次迭代中分别执行相同的 Map 和 Reduce 操作就可以完成。

首先随机选择 *k* 个样本作为中心点，并将这 *k* 个中心点存储在 HDFS 上的一个文件中，作为全局变量。接下来每次迭代由 3 部分组成：Map 函数、Combine 函数和 Reduce 函数。

2.1 Map 函数的设计

Map 函数输入的〈*key*，*value*〉对是 MapReduce 框架默认的格式，即 *key* 是当前样本相对于输入数据文件起始点的偏移量，*value* 是当前样本的各维坐标值组成的字符串。首先，从 *value* 中解析出当前样本各维的值；然后计算其与 *k* 个中心点的距离，找出距离最近的聚簇的下标；最后输出〈*key′*，*va-lue′*〉，其中 *key′* 是距离最近的聚簇的下标，*value′* 是当前样本的各维坐标组成的字符串。函数的伪码为：

```
map(〈key,value〉,〈key′,value′〉)
{
  从 value 中解析出样本对象，记作 instance；
  辅助变量 minDis 初始化为可能的最大值；
  index 初始化为－1；
  For i=0 to k－1 do {
      dis=instance 与第 i 个中心点的距离；
      if dis 小于 minDis {
          minDis=dis；
          index=i；
      }
  }
  将 index 作为 key′；
  将各维坐标值作为 value′；
  输出〈key′,value′〉；
}
```

为了减少算法迭代过程中传输的数据量和通讯代价，在 Map 操作之后，PKMeans 算法中设计一个 Combine 的操作，将每个 Map 函数处理完后的输出数据进行本地合并。因为每个 Map 操作后输出的数据，总是先存储在本地的节点，所以每个 Combine 操作都是在本地执行，通信代价很小。

2.2 Combine 函数的设计

Combine 函数输入的〈*key*，*V*〉对中，*key* 是聚簇的下标，*V* 是分配给下标为 *key* 的聚簇的每个样本的各维坐标值组成的字符串链表。首先从字符串链表中依次解析出每个样本的各维坐标值，并将每一维对应的坐标值分别相加，同时记录下链表中样本的总数。输出的〈*key′*，*value′*〉对中 *key′* 是聚簇的下标；*value′* 是字符串，包括两部分信息：样本总数和各维坐标值的累加和组成的字符串。函数伪码为：

```
combine(〈key,V〉,〈key′,value′〉)
{
  初始化一个数组，用于存储各维坐标的累加值，每个分量初始值为
  0；
  初始化变量 num，记录分配给相同聚簇的样本个数，初始值为 0；
  While (V. hasNext ()) {
      从 V. next()中解析出一个样本的各维坐标值；
      将各维坐标值累加到数组相应的分量中；
      num＋＋；
  }
  将 key 作为 key′；
  构造一个字符串，包含 num 和数组各个分量的信息，将该字符串作
  为 value′；
  输出〈key′,value′〉；
}
```

2.3 Reduce 函数的设计

Reduce 函数输入的〈*key*，*V*〉中，*key* 是聚簇的下标，*V* 是

从各个 Combine 函数传输的中间结果。在 Reduce 函数中首先解析出从每个 Combine 中处理的样本个数和相应节点各维的坐标累加值；然后将对应的各维累加值分别对应相加，再除以总的样本个数，即得新的中心点坐标。函数伪码为：

```
reduce(⟨key,V⟩,⟨key′,value′⟩)
{
  初始化一个数组，用于存储各维坐标的累加值，每个分量初始值为0；
  初始化变量 NUM，记录分配给相同聚簇的总的样本个数，初始值为 0；
  While (V.hasNext ()) {
      从 V. next( ) 中解析出一个样本的各维坐标值和样本个数 num；
      将各维坐标值累加到数组相应的分量中；
      NUM+=num；
  }
  将数组中的每个分量除以 NUM，得到新的中心点坐标；
  将 key 作为 key′；
  构造一个字符串，包含新的中心点各维坐标值的信息，将该字符串作为 value′；
  输出⟨key′,value′⟩；
}
```

根据 Reduce 的输出结果，得到新的中心点坐标，并更新到 HDFS 上的文件中，然后进行下一次迭代，直到算法收敛。

3 实验与结果分析

3.1 实验环境、数据集和评价指标

本文中所有的实验都是在我们实验室搭建的 Hadoop 平台上运行的。平台由 10 台机器、32 核(2 核×4+4 核×6)构成。其中，有 4 台机器是双核 2.8G，4GB 内存；有 6 台机器是四核 2.33G，8G 内存。Hadoop 版本是 0.17.0，java 版本是 1.5.0-14。每台机器之间用千兆以太网卡，通过交换机连接。

实验所用的数据是人工数据，维度是 48 维。为了测试算法的性能，实验中构造了 1G，2G，4G，8G，16G 和 32G 等 6 个不同大小的数据集。

在实验中，采用加速比(speedup)、扩展率(scaleup)和数据伸缩率(sizeup)[8]作为评价指标。

3.2 实验结果

由于 PKMeans 算法中有随机初始化中心点的操作，因此对每一组实验重复执行 20 次，取其平均执行时间作为最终每组实验的结果。

在实验中，我们测试了算法 PKMeans 处理不同大小数据集时的加速比，实验结果如图 1 所示。

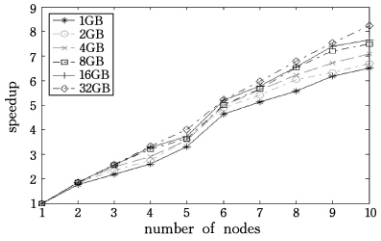

图 1 加速比性能测试结果

从图 1 中可以看出，PKMeans 算法的加速比是接近线性的。并且，随着数据集规模的增大，算法的加速比性能会越来越好。原因有两个：1) PKMeans 算法的 Map 和 Reduce 中设计的⟨key，value⟩对比较合理，使算法能够高效、快捷地实现和执行；2) 在并行算法设计中，增加了 Combine 的操作，使主节点和从节点之间的通讯代价大幅度减少，并且数据集规模越大，通讯量减少的比例越高。因此，当数据集规模越大时，算法的加速比性能越好。

我们用 3 组实验测试 PKMeans 算法的扩展率。第一组是测试不同大小的数据集 1G，2G，4G 和 8G 分别在 1 节点、2 节点、4 节点和 8 节点上的运行效率；第二组是测试不同大小的数据集 2G，4G，8G 和 16G 分别在 1 节点、2 节点、4 节点和 8 节点上的运行效率；第三组是测试不同大小的数据集 4G，8G，16G 和 32G 分别在 1 节点、2 节点、4 节点和 8 节点上的运行效率。实验结果如图 2 所示。

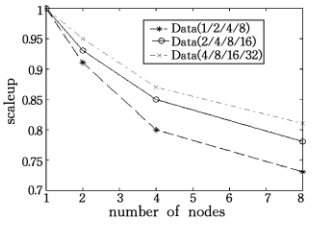

图 2 扩展率性能测试结果

从图 2 中可以看出，对于同一组数据集，当平台节点个数和测试数据集大小同比例增长时，PKMeans 算法的扩展率是逐渐减小的。这是因为，当平台节点个数增加时，节点之间的通讯代价会逐渐增大。当数据规模随节点个数同比例增大时，算法的执行时间会增加。但是，从结果中我们可以看到，随着数据集规模的逐渐增大，PKMeans 算法的扩展率性能越来越好。这是因为，当数据规模增大时，更能发挥每个节点全部的计算能力。并且，在对加速比的分析中指出，PKMeans 中增加了 Combine 的过程，数据集规模越大，主节点和从节点之间通讯代价减少的比例就越高。因此，PKMeans 算法在第二组数据集上的性能优于第一组数据集，在第三组数据集上的性能优于第二组数据集。

图 3 给出了数据伸缩率的性能测试结果。在实验中，分别测试了在不同节点个数的平台下算法的数据伸缩率性能。从结果中可以看出，当平台节点个数少于 3 时，算法的执行时间几乎随数据集的规模同比例增长。随着平台中节点个数的不断增加，算法处理大数据集的效率越来越高。例如，在 10 个节点平台下，算法处理 32G 数据的执行时间才是处理 1G 数据时的 10 倍。实验说明，PKMeans 算法适合运行于大规模的云计算平台，并可以有效地应用于实际中海量数据的分析和挖掘。

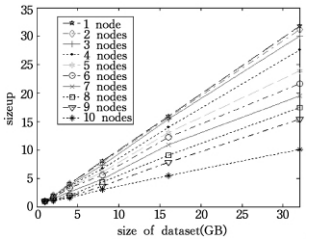

图 3 数据伸缩率性能测试结果

结束语 本文对基于云计算平台 Hadoop 的并行 k-means 算法设计进行了深入的研究。首先简要介绍了 Hadoop 平台的基本组成，包括 HDFS 框架和 MapReduce 各个

(下转第 176 页)

Values:none)进行处理,当其缺失率分别为 10%,20%,30%,40%,50%,60%,70%,80%,90%时,利用两算法分别对数据集进行分类,得出如图 1 所示的实验结果。

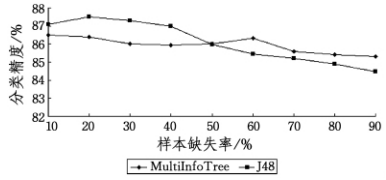

图 1 算法 J48、MultiInfoTree 在不同样本缺失率下分类精度比较

2.2 结果分析

比较表 1 中两种算法分类结果可以看出,分类所得结果有明显不同,原因在于 MultiInfoTree 算法和 J48 算法对属性进行分割的标准不同,它们分别选用联合熵和 Shannon 熵作为分割标准。改进后的算法 MultiInfoTree 效率提高 10～20 倍。其原因是联合熵的计算比 Shannon 熵要简单得多,避免了计算对数 log 消耗的时间,改进算法的执行效率明显优于 C4.5。

从图 1 的实验结果可以看出,随着数据样本缺失值的增加,J48 算法的分类精度下降明显,而 MultiInfotree 算法的分类精度未有很大变化。这说明 J48 算法在处理属性值遗漏数据时会带入偏置,而 MultiInfotree 算法在生成决策树的过程中能够消除值遗漏数据对测试属性选择的影响。当缺失率大于 50%时,改进后的算法的分类精度提高最为明显,这说明 MultiInfoTree 算法较适合于缺失率在这个区间的数据集。

结束语 在实际的数据中,属性值遗漏数据是无处不在的。本文把基于联合熵的信息增益率作为决策树测试属性选择的标准,它能够在生成决策树的过程中消除值遗漏数据对测试属性选择的影响,更适合于实际数据。最后,通过实验数据验证了 MultiInfoTree 算法能够从总体上提高算法执行效率和分类精度,非常适合于样本缺失率大于 50%的数据集的分类问题。当样本缺失率在其他区间时,将通过组合分类器[10]的技术来提高分类准确率。我们会在以后的工作中不断扩展和细化这个领域的研究,使之更加完善。本文研究成果解决了从大规模、不确定性数据集中发现决策树分类模型的问题,为应用决策树分类技术提供了更为广阔的空间。

参 考 文 献

[1] Gustavo E A, Batista P A, Monard M C. An Analysis of Four Missing Data Treatment Methods for Supervised Learning [J]. Applied Artificial Intelligence, 2003, 17(5/6): 519-533

[2] Kryszkiewicz M. Rules in incomplete information systems [J]. Information Sciences, 1999, 113: 271-292

[3] Mingers J. An empirical comparison of selection measures for decision-tree induction [J]. Machine Learning, 1989, 3(4): 319-342

[4] Safavian S R, Landgrebe D. A Survey of Decision Tree Classifier Methodology[R]. 47907. School of Electrical Engineering, Purdue University, 1991: 1-58

[5] 冯少荣. 决策树算法的研究与改进[J]. 厦门大学学报:自然科学版, 2007, 20 (4): 498-500

[6] Quinlan J R. C4.5: Programs for Machine Learning [S]. Morgan Kaufman, 1993

[7] Qian Yunhua, Liang Jiye. A new method for measuring the uncertainty in incomplete information systems [J]. International Journal of Uncertainty. Fuzziness and Knowledge-Based Systems, 2008(9)

[8] Leung Y, Li D Y. Maximal consistent block technique for rule acquisition in incomplete information systems [J]. Information Science, 2003, 153: 85-106

[9] 赵蕊. 基于 WEKA 平台的决策树算法设计与实现[D]. 长沙:中南大学, 2007: 43-46

[10] 旷海兰, 罗可, 刘新华, 等. 一种基于粗糙集理论的组合分类器构造方法[J]. 计算机工程与应用, 2006, 16

(上接第 168 页)

阶段的工作流程以及结构关系。然后,给出基于 Hadoop 的并行 k-means 算法设计时需要思考的主要问题、算法设计的主要流程以及方法和策略等。最后,通过在多组不同大小数据集上的实验表明,我们设计的并行聚类算法 PKMeans 适合运行于大规模云计算平台,可以有效地应用于实际中海量数据的分析和挖掘。

随着云计算概念的兴起,基于云计算平台的数据挖掘、聚类算法的研究逐渐成为国内外学者的研究热点。未来的研究方向包括:1) 研究聚类算法并行化的一般规律,找到数据规模、算法复杂性、节点数之间的关系,发现加速比和可扩展性的影响因素,从而设计出高效的并行聚类算法;2) 研究基于云计算平台的数据挖掘应用中的信息安全和隐私保护等问题,该问题的解决对于云计算在实际商务中的应用将起到关键性的作用。

参 考 文 献

[1] Han J W, Kamber M. Data mining: concepts and techniques [M]. San Francisco, US: Morgan Kaufmann, 2001

[2] Buyya R, Yeo C S, Venugopal S. Market-oriented cloud computing: vision, hype, and reality for delivering IT services as computing utilities, Keynote Paper[C] // Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications. Dalian, China, 2009: 25-27

[3] Armbrust M, Fox A. Above the clouds: a Berkeley view of cloud computing[R]. USA: University of California at Berkeley, 2009

[4] Erdogmus H. Cloud computing: does nirvana hide behind the nebula[J]. IEEE Software, 2009, 26(2): 4-6

[5] 郑纬民. 云计算的大幕已经拉开[J]. 中国计算机学会通讯, 2009, 2(6): 6-7

[6] Ghemawat S, Gobioff H, Leung S. The google file system[J]. S ACM SIGOPS Operating Systems Review, 2003, 37(5): 29-43

[7] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[C] // Proceedings of Operating Systems Design and Implementation. San Francisco, CA, 2004: 137-150

[8] Xu X W, Jager J, Kriegel H P. A fast parallel clustering algorithm for large spatial databases[J]. Data Mining and Knowledge Discovery, 1999, 3(3): 263-290