# Spark 平台下聚类算法的性能比较

海 沫[1,2] 张 游[3]
(中央财经大学信息学院 北京 100081)[1]
(电子科技大学网络与数据安全四川省重点实验室 成都 610054)[2]
(卡内基梅隆大学海因茨学院信息系统管理系 匹兹堡 999039)[3]

**摘 要** 通过实验,从运行时间、加速比、可扩展性和规模增长性 4 个方面比较了 Spark 平台中 3 种典型的聚类算法即 K-means 聚类算法、二分 K-means 聚类算法和高斯混合聚类算法的性能。实验结果表明:1)随着节点个数的增加,3 种算法对百兆以上规模数据集聚类的运行时间明显减少;2)当数据集规模大于 500MB 时,3 种算法的加速比均有明显提高,且随着节点个数的增加,加速比近似于线性增长;3)3 种算法的可扩展性随着节点个数的增加而降低,当数据集规模大于 500MB 时,相对于 K-means 和高斯混合算法,二分 K-means 算法的可扩展性最差;4)当数据集规模大于 100MB 时,高斯混合算法的规模增长性远高于 K-means 和二分 K-means 算法。

**关键词** Spark,K-means 聚类,二分 K-means 聚类,高斯混合聚类,运行时间,加速比,可扩展性,规模增长性
**中图法分类号** TP311 **文献标识码** A

## Performance Comparison of Clustering Algorithms in Spark

HAI Mo[1,2] ZHANG You[3]
(School of Information, Central University of Finance and Economics, Beijing 100081, China)[1]
(Network and Data Security Key Laboratory of Sichuan Province, University of Electronic Science and Technology of China, Chengdu 610054, China)[2]
(School of Information Systems & Management, Heinz College, Carnegie Mellon University, Pittsburgh 999039, USA)[3]

**Abstract** The performance of three typical clustering algorithms which are K-means, Bisecting K-means and Gaussian Mixture in Spark, were compared by the experiments from runtime, speedup, scalability and size up. The results show that when the scale of the dataset is hundreds of megabytes, as the number of nodes increases, the runtime of the three algorithms decreases more obviously. When the scale of the dataset is larger than 500MB, the speedup of the three algorithms increases more obviously, and the speedup increases linearly with the increase of the number of nodes. The scalability of the three algorithms decreases with the increase of the number of nodes. When the scale of the dataset is larger than 500MB, the scalability of the Bisecting K-means algorithm is the lowest compared to that of the K-means and Gaussian Mixture algorithm. When the scale of the dataset is larger than 100MB, the sizeup of the Gaussian Mixture algorithm is much larger than that of K-means algorithm and bisecting K-mean algorithm.

**Keywords** Spark, K-means clustering, Bisecting K-means clustering, Gaussian mixture clustering, Runtime, Speedup, Scalability, Sizeup

## 1 引言

随着互联网和移动互联网技术的飞速发展,各种 Web 应用和手机应用所产生的数据量呈爆炸性增长,如何高效地对这些海量数据进行聚类已成为当今的研究热点。若采用传统的单机方式,无论是内存还是计算能力,都无法满足对大数据进行聚类分析的需求,而将分布式框架与聚类算法相结合为解决大规模数据聚类问题提供了一种有效的途径。

Hadoop[1-3] 是一种基于 MapReduce[4-5] 计算模型的分布式计算框架。MapReduce 不能充分地利用分布式内存,在进行迭代计算时,存在代价很高的中间结果的磁盘读写和资源申请过程,效率很低,因而对需要在多个并行操作之间重用中间结果的应用效率不高,而 Spark[6-7] 的出现刚好解决了这个问题。Spark 是 UC Berkeley AMPLab 于 2009 年发起的,类似于 Hadoop MapReduce 的分布式计算框架。Spark 是用 Scala 语言编写,并且基于弹性分布式数据集的分布式计算框架。与 Hadoop 相比,Spark 允许将 MapReduce 的中间结果存储在内存中,从而省去了大量的磁盘 I/O 操作,在进行迭代计算时效率更高,因而适用于在多个并行操作间重用中间结果的情形,如迭代式算法、交互式数据分析、流应用。

作为一种迭代式的算法,聚类在 Spark 上实现的效率优于 Hadoop。基于 Spark 的机器学习库 MLlib 实现了各种经

本文受网络与数据安全四川省重点实验室开放课题(NDSMS201604),中央财经大学青年教师发展基金项目(QJJ1634)资助。
海 沫(1978-),女,博士,副教授,CCF 高级会员,主要研究方向为分布式系统、大数据分析和处理,E-mail:haimo_hm@163.com;张 游(1995-),男,硕士,主要研究方向为机器学习、大数据挖掘。

典的聚类算法。本文选取了 MLlib 中 3 种有代表性的聚类算法即 K-means、二分 K-means 和高斯混合算法进行性能比较。在不同类型及不同规模的数据集、不同规模的集群下，通过设置不同的算法参数来比较这些算法的性能。

## 2　相关工作

梁彦[8]开发了基于 Spark 平台的并行 K-means 算法和并行 Canopy K-means 算法，并从在准确性、加速比、可扩展性方面对这两种算法进行了比较。实验结果表明：算法并行化后具有较好的聚类结果，在面对海量数据时有较好的加速比和可扩展性。与 Hadoop 平台比较，基于 Spark 平台的算法的并行化效率更高。唐振坤[9]针对机器学习任务中的常见场景，基于 Spark 平台设计和实现了并行 K-means 聚类算法以及并行数据流 K-means 聚类算法，并通过实验验证了这些算法在运行时间、加速比、吞吐量 3 个方面都优于串行算法和基于 Hadoop 平台实现的算法。陈虹君[10]针对 Spark MLlib 中的 K-means 算法进行了研究，对算法的实现原理进行了分析，并在 3 种不同的数据集上对 K-means 算法进行了测试，分析了其应用场景和缺点。王桂兰等[11]提出了 Spark 平台下的模糊 C 均值算法——Spark FCM，实验结果表明该算法具有较好的可扩展性，并能适应大规模数据集，其性能与数据量呈线性关系，并且其在集群环境下的性能比单机提高 2～3 倍。吴哲夫等[12]针对 K-means 算法在聚类过程中初始值选取的随机性问题，基于非均匀采样原则对该算法进行了改进，并基于 Spark 平台对改进算法进行了并行实现，实验结果表明该改进算法在处理海量数据集时具有更高的准确性和稳定性，且具有较好的加速比和可扩展性。

综上所述，目前对 Spark 平台下聚类算法的研究可分为两类：1）在 Spark 上实现某种聚类算法后，研究其性能；2）对 Spark MLlib 中的某种聚类算法的性能进行研究。但已有研究缺乏对 Spark MLlib 中已有的多种聚类算法的性能比较。本文通过对 Spark MLlib 中 3 种代表性的聚类算法（K-means 聚类算法、二分 K-means 聚类算法和高斯混合聚类算法）进行性能比较，以方便基于 Spark MLlib 开发聚类应用的开发者选取合适的聚类算法。

## 3　算法介绍

### 3.1　K-means 聚类算法

K-means 聚类算法是一种简单、收敛速度快且易于实现的经典聚类算法。该算法需要事先指定聚类簇个数 $k$，并随机选取 $k$ 个数据点作为初始聚类中心；然后计算数据集中每一个数据点到各中心的距离，并根据最近距离原则将各数据点划分给最近的簇，直到所有数据点计算完成。通过迭代不断调整聚类中心，直到收敛条件满足。这里的收敛条件可以是达到最大的迭代次数、相邻两次迭代中心点没有改变或者是达到准则函数收敛条件。

### 3.2　二分 K-means 聚类算法

二分 K-means 聚类算法首先将所有点作为一个簇，然后将该簇一分为二，之后选择能最大程度降低聚类代价函数（也就是误差平方和）的簇，将其划分为两个簇，以此进行下去，直到簇的数目等于用户给定的数目 $k$ 为止。其中隐含的原则是：由于聚类的误差平方和能够衡量聚类性能，该值越小则表示数据点越接近于它们的质心，聚类效果就越好，因而需要对误差平方和最大的簇再次进行划分。因为误差平方和越大，表示该簇聚类越不好，越有可能是多个簇被当成了一个簇，所以首先需要对这个簇进行划分。该算法在一定程度上解决了 K-means 算法对初始聚类中心敏感的问题，并且由于减小了相似度计算，执行速度比 K-means 算法更快。

### 3.3　高斯混合聚类算法

高斯混合聚类算法是一种基于高斯混合模型的软聚类算法，通过计算出每个数据点属于每个类的后验概率，将数据点划分到后验概率最大的类。高斯混合模型基于多变量正态分布对样本的概率密度分布进行估计，而估计采用的训练模型是几种高斯模型的加权和。每种高斯模型代表一个类。对样本中的数据分别在不同高斯模型上投影，就会分别得到在每个类上的概率，选取概率最大的类加入。与 K-means 聚类算法相似，高斯混合模型也使用迭代算法计算，最终收敛到局部最优，但与 K-means 聚类算法相比，该模型更适合各类大小不同、聚类间有相关关系的聚类。与模糊 K-means 聚类算法类似，高斯混合聚类算法也是计算出每个数据点属于每个类的概率，但模糊 K-means 聚类算法假设各聚类形状近似球形，并且大小也近似相等，这相当于所有成分协方差矩阵相同的高斯混合分布；而高斯混合分布函数允许设置各成分不同的协方差，默认情况下是为每个成分估计一个分离的无约束的协方差矩阵。

## 4　性能比较

### 4.1　实验环境

实验环境为 3 台 Windows Server 2012 服务器，并在每台服务器上安装 2 个 Linux 虚拟机，以此为基础构建由 6 个节点构成的集群进行实验。每个节点的具体配置如表 1 所列。

表 1　Spark 集群节点配置情况

| 名称 | 配置 |
|---|---|
| CPU | Intel 至强 E5 2609 2.40GHz |
| 内存 | 2GB |
| 硬盘 | 20GB |
| Linux 版本 | Red Hat Enterprise Linux server 6.2 |
| JDK 版本 | JDK1.7.0_79 |
| SCALA 版本 | Scala-2.10.6 |
| HADOOP 版本 | Hadoop-2.6.0 |
| SPARK 版本 | Spark-1.6.1 |

Spark 集群下的节点与 Hadoop 类似，分为 master 节点和 worker 节点；同时，Spark 通过与 HDFS 交互来读写文件。Spark 支持 3 种部署方式：Standalone 模式、Spark on YARN 模式和 Spark on Mesos 模式，实验采用了 Standalone 模式。运行过程中，master 节点作为主节点完成作业的调度并监控各 worker 节点的运行状况，worker 节点完成 master 节点分配的任务。

### 4.2　数据集

实验数据来自于 UCI 机器学习库中的 synthetic_control 数据集，这是一个时间序列数据，可用来对分类和聚类算法的性能进行测试。该数据集包含 600 个数据点，每个数据点包含 60 个维度的值。由于原始数据集的大小只有 287kB，使用随机复制的方法对原始数据集进行扩展，分别生成了包含 3000，30000，300000，1200000，3000000 个数据点的实验数据集，其数据量和数据规模如表 2 所列。

表 2 测试数据集

| 数据集名称 | 数据点个数 | 数据集规模 |
|---|---|---|
| sc(synthetic_control) | 600 | 277kB |
| sc*5 | 3000 | 1.35MB |
| sc*50 | 30000 | 13.5MB |
| sc*500 | 300000 | 135.4MB |
| sc*2000 | 1200000 | 541.6MB |
| sc*5000 | 3000000 | 1.32GB |

### 4.3 性能评价指标

实验采用了运行时间、加速比、可扩展性和规模增长性作为评价指标。

(1)运行时间

运行时间是最直观地表示聚类算法执行效率的指标。聚类算法所需的执行时间越短,其执行效率越高。

(2)加速比

加速比是在单机环境和并行环境中运行同一任务消耗的时间的比率,用来衡量并行系统并行化的性能和效果。在保持测试数据集不变的情况下,通过改变节点个数来计算加速比。其计算方式如式(1)所示:

$$Speedup(m)=\frac{T_1}{T_m} \tag{1}$$

其中,$T_1$ 表示在一个节点上运行某一算法的执行时间,$T_m$ 表示 $m$ 个节点在同一数据集上运行同一算法的执行时间。如果加速比随着节点个数的增加而保持线性增长,则表示多个节点能有效缩短算法的运行时间。然而,线性的加速比是难以实现的,因为随着节点个数的增加,会引入额外的通信开销。

(3)可扩展性

可扩展性是评价算法性能随着节点个数的增加而按比例提高的能力。其计算方式如式(2)所示:

$$Scaleup(m)=\frac{Speedup(m)}{m} \tag{2}$$

其中,$m$ 表示节点个数,$Speedup(m)$ 表示 $m$ 个节点上的加速比。增加节点个数会增加额外的通信开销并降低每个节点的利用率,因此对于某个特定的并行算法,其能否有效利用不断增加的节点个数的能力应是受限的,而可扩展性指标就用于度量并行算法能否有效利用可扩展的节点个数的能力。

(4)规模增长性

规模增长性反映了随着数据量的增加,并行算法性能的变化情况。在保持节点个数不变的情况下,通过改变数据集规模来计算规模增长性。其计算方式如式(3)所示:

$$Sizeup(m)=\frac{T_{mDB}}{T_{DB}} \tag{3}$$

其中,$T_{DB}$ 表示在 DB 数据集上运行算法的执行时间;$T_{mDB}$ 表示在节点个数不变时,在 $m$ 倍 DB 数据集上运行同一算法的执行时间。

### 4.4 实验结果

#### 4.4.1 运行时间比较

当数据集为原始的 synthetic_control 数据集、sc*5、sc*50、sc*500、sc*2000 时,在节点个数分别为 1,2,4 和 6 的情况下对 K-means、二分 K-means 和高斯混合 3 种算法的运行时间进行比较,其结果如图 1—图 5 所示。从图中可以发现,对于百兆以下规模数据集,随着节点个数的增加,运行时间减少的幅度不大,甚至在某些情况下节点个数的增加会导致运行时间的增加,这是由于小规模数据量下算法用于迭代计算的时间开销很小,大部分的时间开销都花在了节点间的通信上。

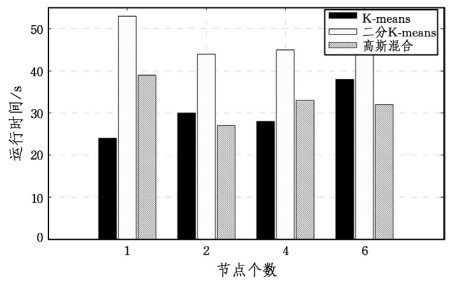


图 1 synthetic_control 数据集下 3 种算法的运行时间比较

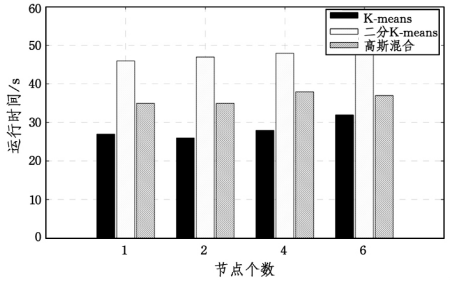


图 2 sc*5 数据集下 3 种算法的运行时间比较

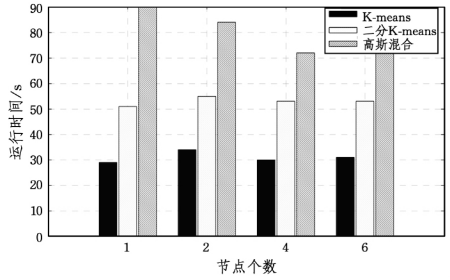


图 3 sc*50 数据集下 3 种算法的运行时间比较

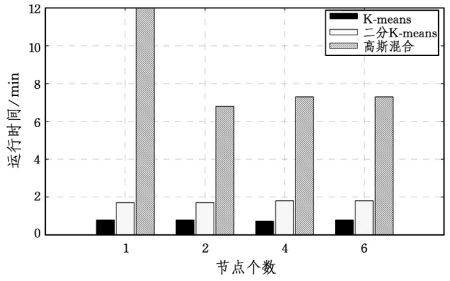


图 4 sc*500 数据集下 3 种算法的运行时间比较

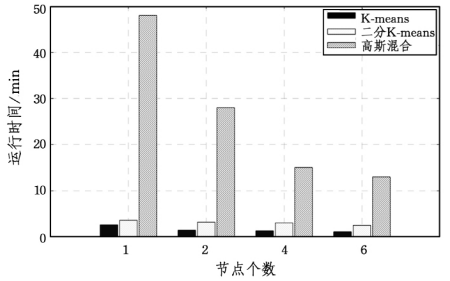


图 5 sc*2000 数据集下 3 种算法的运行时间比较

通过对实验结果进行分析,可以发现:

(1)对于 K-means 算法,当节点个数为 2 时,其在数据集 sc*2000 下的运行时间约为在数据集 synthetic_control 下的 2.8倍;当节点个数为 4 时,其在数据集 sc*2000 下的运行时间约为在数据集 synthetic_control 下的 2.79 倍;而当节点个数为 6 时,其在数据集 sc*2000 下的运行时间约为在数据集 synthetic_control 下的 1.7 倍。

(2)对于二分 K-means 算法，当节点个数为 2 时，其在数据集 sc＊2000 下的运行时间约为在数据集 synthetic_control 下的 4.4 倍；当节点个数为 4 时，其在数据集 sc＊2000 下的运行时间约为在数据集 synthetic_control 下的 4 倍；而当节点个数为 6 时，其在数据集 sc＊2000 下的运行时间约为在数据集 synthetic_control 下的 3.2 倍。

(3)对于高斯混合算法，当节点个数为 2 时，其在数据集 sc＊2000 下的运行时间约为在数据集 synthetic_control 下的 62.2 倍；当节点个数为 4 时，其在数据集 sc＊2000 下的运行时间约为在数据集 synthetic_control 下的 27.3 倍；而当节点个数为 6 时，其在数据集 sc＊2000 下的运行时间约为在数据集 synthetic_control 下的 24.4 倍。

以上结果表明：随着节点个数的增加，3 种算法对百兆以上规模数据集的聚类效率的提高越发明显。

4.4.2　加速比比较

当数据集为原始的 synthetic_control 数据集、sc＊5、sc＊50、sc＊500、sc＊2000 时，在节点个数分别为 1，2，4 和 6 的情况下对 K-means、二分 K-means 和高斯混合 3 种算法的加速比进行比较，其结果如图 6－图 10 所示。

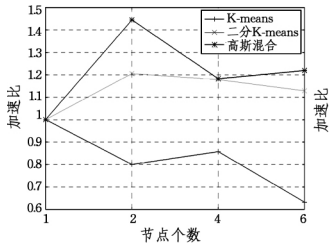


图 6　synthetic_control 数据集下 3 种算法的加速比比较

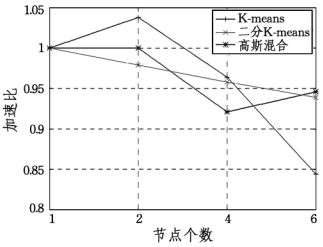


图 7　sc＊5 数据集下 3 种算法的加速比比较

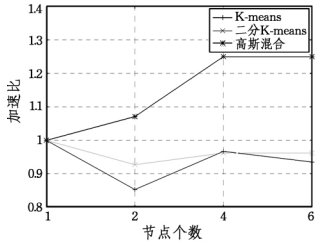


图 8　sc＊50 数据集下 3 种算法的加速比比较

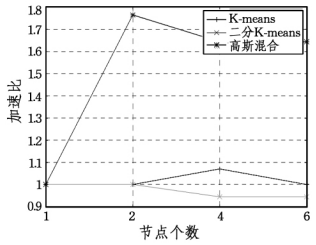


图 9　sc＊500 数据集下 3 种算法的加速比比较

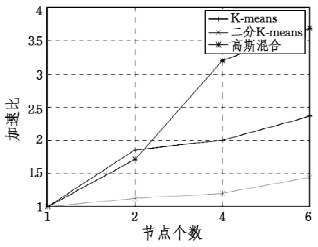


图 10　sc＊2000 数据集下 3 种算法的加速比比较

从以上图中可以观察到：当数据集规模小于 500MB 时，随着节点个数的增多，3 种算法的加速比并没有呈线性增长；而当数据集规模大于 500MB 时，随着节点个数的增多，3 种算法的加速比呈现近似线性增长的趋势。这表明：当数据规模到达 500MB 数量级时，3 种算法的加速比随着节点个数的增加而近似呈线性增长。

4.4.3　可扩展性比较

当数据集为原始的 synthetic_control 数据集、sc＊5、sc＊500、sc＊2000 时，在节点个数分别为 1，2，4 和 6 的情况下对 K-means、二分 K-means 和高斯混合 3 种算法的可扩展性进行比较，结果如图 11－图 14 所示。从图中可观察到：随着节点个数的增加，3 种算法的可扩展性均逐渐减小；当数据集规模大于 500MB 时，相对于 K-means 和高斯混合算法，二分 K-means 算法的可扩展性最低，因而对于 500MB 规模以上的数据集，相比于二分 K-means 算法，K-means 和高斯混合算法更适合在集群环境下运行。

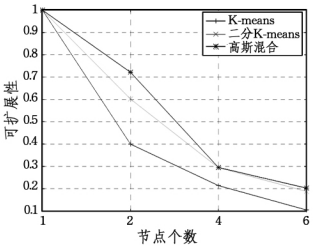


图 11　synthetic_control 数据集下 3 种算法的可扩展性比较

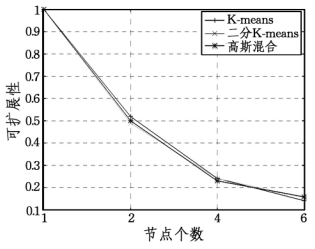


图 12　sc＊5 数据集下 3 种算法的可扩展性比较

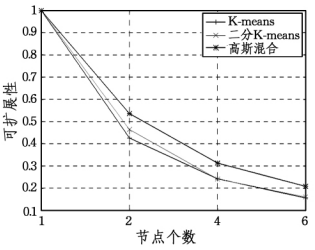


图 13　sc＊500 数据集下 3 种算法的可扩展性比较

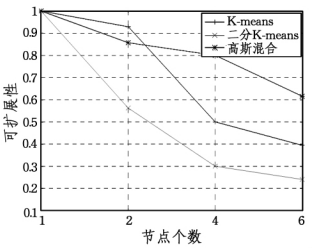


图 14　sc＊2000 数据集下 3 种算法的可扩展性比较

4.4.4　规模增长性比较

当节点个数分别为 2，4 和 6 时，在数据集分别为原始的 synthetic_control 数据集、sc＊5、sc＊50、sc＊500、sc＊2000 的情况下，对 K-means、二分 K-means 和高斯混合 3 种算法的规模增长性进行比较，结果如图 15－图 17 所示。

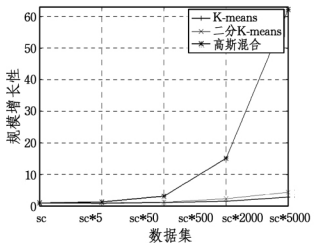


图 15　节点个数为 2 时 3 种算法的规模增长性比较

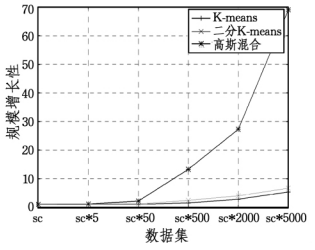


图 16　节点个数为 4 时 3 种算法的规模增长性比较

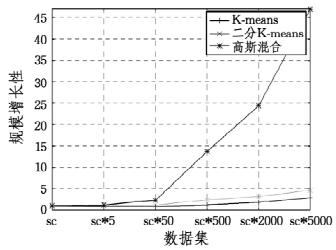


图 17　节点个数为 6 时 3 种算法的规模增长性比较

从以上图中可观察到：当数据集规模大于 100MB 时，高斯混合算法的规模增长性远大于 K-means 和二分 K-means 算法，这表明 K-means 和二分 K-means 算法对 100MB 以上规模的数据集的聚类效率更高；当数据集规模大于 1GB 时，

二分K-means算法在6个节点下的规模增长性约为4个节点下的69%，且高斯混合算法在6个节点下的规模增长性约为4个节点下的68%，节点个数比例都近似于4/6，说明在该规模数据集下集群中节点的计算能力能得到充分利用。因而，对1GB以上规模的数据集进行聚类更适合集群结构。

**结束语** 通过对Spark中的3种代表性聚类算法即K-means、二分K-means和高斯混合聚类算法的性能进行比较，可以发现：1)当数据集规模小于500MB时，随着节点个数的增多，3种算法的加速比并没有呈现线性增长的趋势；而当数据集规模大于500MB时，3种算法的加速比均有了明显提高，且随着节点个数的增加，加速比近似于线性增长，因此，对500MB以上规模的数据集进行聚类适合采用集群结构来提高3种算法的运行效率。2)3种算法的可扩展性随着节点个数的增加而降低。当数据集规模大于500MB时，相对于K-means和高斯混合算法，二分K-means算法的可扩展性最低，因而，对于500MB以上规模数据集，K-means和高斯混合算法更适合在集群环境下运行。3)当数据集规模大于100MB时，高斯混合算法的规模增长性远高于K-means和二分K-means算法，这表明K-means和二分K-means算法对100MB以上规模的数据集的聚类效率更高。

## 参 考 文 献


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

[2] 周品. Hadoop云计算实战[M]. 北京：清华大学出版社，2012.

[3] KONSTANTIN S. The Hadoop distributed file system [C]// The 26th Symposium on Mass Storage Systems and Technologies. IEEE，2010：1-10.

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

[5] DEAN J，GHEMAWAT S. MapReduce：a flexible data processing tool[J]. Communications of the ACM，2010，53(1)：72-77.

[6] KARAU H. Fast Data Processing With Spark[M]. Bermingham：Packt Publishing Ltd，2013.

[7] ZAHARIA M，CHOWDHURY M，DAS T，et al. Fast and interactive analytics over Hadoop data with Spark[J].USENIX，2012，37(4)：45-51.

[8] 梁彦. 基于分布式平台Spark和YARN的数据挖掘算法的并行化研究[D]. 广州：中山大学，2014.

[9] 唐振坤. 基于Spark的机器学习平台设计与实现[D]. 福州：厦门大学，2014.

[10] 陈虹君. 基于Spark框架的聚类算法研究[J]. 电脑知识与技术，2015，11(4)：56-57，60.

[11] 王桂兰，周国亮，萨初日拉，等. Spark环境下的并行模糊C均值聚类算法[J]. 计算机应用，2016，36(2)：342-347.

[12] 吴哲夫，张彤，肖鹰. 基于Spark平台的K-means聚类算法改进及并行化实现[J]. 互联网天地，2016(1)：44-50.


(上接第389页)

引出了TrustedMarket可信软件市场模型，对模型中的节点、服务节点、关系成熟度、忠诚度、推荐信任值等重要概念进行了定义，并详细研究了模型中关系成熟度和忠诚度的计算、初始信任值指定和信任值计算的关键技术。介绍了TrustedMarket模型在Android操作系统平台的智能终端上进行实验验证的过程。讨论了实验环境设置、实验详细过程等问题，最后对实验结论进行了深入分析。实验过程和结果分析指出，TrustedMarket模型具有较高的可用性，可解决目前智能终端应用软件市场“不可信”的问题。

下一步，可以基于本文提出的可信软件市场模型构建安全性要求高的专用领域Android可信软件市场，保证专用领域Android智能终端应用软件来源可信。

## 参 考 文 献


[1] BETH T，BORCHERDING M，KLEIN B. Valuation of trust in open network[C]// Proceedings of the European Symposium on Research in Security (ESORICS). Brighton：Springer-Verlag，1994.

[2] RD T，KNAPSKOG S J. A metric for trusted systems. Global IT Security[C]// Proceedings of the 21st Nationed Security Conference. NSA，1998：87-91.

[3] ABDUL-RAHMAN A，HAILES S. A distributed trust model [C]// Proceedings of the 1997 workshop on New Security Paradigm. NSPW，1997.

[4] MUI L，MOHTASHEMI M，HALBERSTADT A. A Computational Model of Trust and Reputation[C]// Proc. 35th Hawaii International Conference on System Sciences. 2002.

[5] ABERER K，DESPOTOVIC Z. Managing Trust in a Peer-2-Peer Information System[C]// Proceedings of the Ninth International Conference on Information and Knowledge Management. 2001.

[6] XIONG L，LIU L. Peer Trust：Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities[C] // Proceedings of 2003 IEEE Conference on E-Commerce. 2003.

[7] 郭晶晶，马建峰，李琦，等. 基于博弈论的移动自组织网络的信任管理方法[J]. 通信学报，2014，35(11)：50-58.

[8] 林伟，郑相涵. 基于私有云信任度量的对等网络节点安全互联模型[J]. 电信科学，2015，31(1)：1-6.

[9] 蒋黎明，刘志明，张琨，等. 基于动态分组的开放分布系统信任度量与管理研究[J]. 通信学报，2015，36(1)：1-7.

[10] 李建，何永忠，徐开勇. 基于信任度量的软件下载服务框架[J]. 武汉大学学报(自然科学版)，2008，33(10)：1062-1066.

[11] ZHENG Y，PENG Z，ATHANASIOS V. Vasilakos. A survey on trust management for Internet of Things[J]. Journal of Network and Computer Applications，2014，42(3)：124-134.

[12] 龚洁中，陈恭亮，李建华. 智能移动终端的信任管理技术研究[J]. 中国电子科学研究院学报，2011，6(6)：567-570.

[13] 林庆国，刘宴兵. 一种基于信任的动态访问控制策略[J]. 重庆邮电大学学报(自然科学版)，2010，22(4)：478-482.

[14] VIRENDRA M，JADLIWALA M，CHANDRASEKARAN M，et al. Quantifying trust in mobile ad-hoc networks[C]// Proc. IEEE International Conf. on Integration Knowledge Intensive Multi-Agent Syst. Waltham，USA，Apr. 2005.

[15] VELLOSO P B，LAUFER R P，CUNHA D D O. et al. Trust Management in Mobile Ad Hoc Networks Using a Scalable Maturity-Based Model[J]. IEEE Transactions on Network and Service Management，2010，7(3)：172-185.