

基于 MapReduce 的 CTK 加权聚类改进算法

王万良, 胡 禹

(浙江工业大学 计算机科学与技术学院, 浙江 杭州 311023)

摘 要: 本文提出了一种基于 MapReduce 的分布式聚类改进算法, 该算法将聚类分两阶段进行, 首先改进了 Canopy 算法, 引入梯度变化来确定初始中心点以及最佳簇数, 解决了传统算法对初始值的依赖性以及对聚类个数的不确定性, 设置了区域半径并动态改变, 避免了聚类过程中的局部最优, 并采用了信息熵加权, 解决了相似度计算的特征权重的问题, 最终结合 MapReduce 分布式计算模型, 设计了算法的并行策略与方案. 试验结果表明该算法在准确性、加速比、扩展性上具有良好的性能。

关键词: 大数据; 聚类算法; Canopy 算法; MapReduce;

中图分类号: TP3

文献标识码: A

文章编号: 1000-7180(2018)12-0105-05

Improved CTK Weighting Clustering Algorithm Based on MapReduce

WANG Wan-liang, HU Yu

(College of Computer Science & Technology, Zhejiang University of Technology, Hangzhou 311023, China)

Abstract: This paper introduces an improved algorithm of Distributed Clustering Based on MapReduce, the process of clustering will be divided into two stages, firstly, introduce Canopy algorithm, find out the suitable K of clustering algorithm by the change of Gradient value. That reduce the number of iterations and avoid the uncertainty of initial center point results in. Then dynamically change the radius of the region to solve the problem of similarity of high-dimensional data sets and solve the problem of characteristic weight of similarity calculation with the weighting of information entropy. Finally, the parallel strategy and scheme of the algorithm are designed according to the MapReduce distributed computing model. Experimental results show that the proposed algorithm has good performance in accuracy, speedup and scalability.

Key words: Big data, Cluster, Canopy algorithm, MapReduce

1 引言

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. 它不需要标签数据等先验知识就可以对数据集进行划分^[1]. 传统的 K-means 算法因其简洁、高效的特性而成为使用最为广泛的聚类算法^[2]. 随着聚类分析的数据体量不断增大, 传统的聚类算法已经较难满足大数据时代的需要. 针对传统聚类算法的不足, 很多学者提出了改进方案.

针对聚类初始值选取问题, 雷小锋等人通过多

次随机采样来确定初始 K 值, 初步解决了聚类中心盲目选取的问题^[3]. Frey 等在 Science 提出了近邻传播算法 (Affinity Propagation Clustering), 但其需要事先计算对象之间的相似度, 并未实现并行化, 迭代的时间复杂度较高^[4]. 另外还有学者提出基于利用智能算法的解决方法, 如 PSO^[5,6], 遗传算法^[7,8]等.

针对海量数据集, 很多学者依托于并行计算框架进行算法研究. Dean 等将传统的聚类算法移植到 MapReduce 框架上. Xie 等根据 MapReduce 框架,

实现了区域以及候选队列的查询和处理,提高了算法的聚类效率^[9]. 翟东海等人提出了采用最大最小距离来确定初始中心点. 赵硕等人提出使用 Canopy 算法来优化聚类,但其初始区域半径大小靠人工来选取,因此效果并不稳定.

基于以上算法研究,本文提出了基于 MapReduce 的 CTK 加权聚类算法. 基本思想如下:

(1)首先,以 Canopy 聚类算法为基础,通过引入梯度变化来确定最佳输出,消除算法对于人工设值以及初始阈值选取的依赖. 最终得到最佳的簇集数以及初始中心点.

(2)在计算相似度时,增加了确定性判定过滤,从而减少比较次数,缩短迭代时间.

(3)在通过确定性判定的数据集合中,计算信息熵加权,确定各属性权重. 之后再进行距离计算并做聚类划分. 通过加权来修正相似度函数,优化聚类效果.

2 相关知识

2.1 Hadoop 以及 MapReduce 并行编程框架

Hadoop 是 Apache 基金会所开发的一种分布式系统编程框架,用于大规模数据集的并行运算. 它将底层实现“透明化”,可以自动地实现资源划分调度、集群容错等功能. Hadoop 可以完美地支持 MapReduce 的任务运行,它将任务分解成 Map 和 Reduce 两个函数阶段,Map 和 Reduce 函数处理的数据均以键值对形式出现. Map 函数根据用户输入的 $\langle \text{Key}_R^{\text{In}}, \text{Value}_R^{\text{In}} \rangle$ 生成中间结果 $\langle \text{Key}_R^{\text{Out}}, \text{Value}_R^{\text{Out}} \rangle$,而 Reduce 函数的作用是将 $\langle \text{Key}_M^{\text{In}}, \text{Value}_M^{\text{In}} \rangle$ 计算生成最终结果 $\langle \text{Key}_M^{\text{Out}}, \text{Value}_M^{\text{Out}} \rangle$. 其中 $\text{Key}_M^{\text{Out}}$ 需要隐式地转为 Key_R^{In} , $\text{Key}_M^{\text{Out}}$ 也需要变换为 Key_R^{In} . 此外用户还可以实现 Combine 函数,用来减少两阶段函数的转移数据量,缓解网络传输的压力. 本文算法是基于 Hadoop 平台的 MapReduce 框架实现的.

2.2 聚类算法

为了解决传统聚类算法迭代速度较慢、簇值需人为设定等不足,出现了 Canopy 聚类算法,它主要被用于海量高维数据的聚类. 与其他传统聚类算法做比较,Canopy 在聚类迭代速度上有较强优势,它无需知道聚类中心的个数就可以粗略地将数据划分成若干个子集. Canopy 算法步骤如下:

Step1: 将数据集 D 进行预处理后得到向量 D' 并写入内存. 指定两个距离阈值: T_1, T_2 , 其中 $T_1 \geq T_2$.

Step2: 从内存数据列表 D 中随机选取一数据点 P 作为初始,并将其作为中心点加入到 Canopy 中心列表 C 中. 并依次计算向量 D 中每个点到所有 Canopy 中心点的相似度距离 ($\text{Distance}(D_i, C_j)$).

Step3: 若 D_i, C_j 距离均大于 T_1 , 则将 D_i 作为一个新的中心加入到集合 C 中. 若距离小于 T_1 , 则将 D_i 加入以 C_j 为中心的 Canopy 中. 若距离小于 T_2 , 则表明 D_i 与 C_j 非常接近, D_i 不再能作为其他 Canopy 的中心,因此将其从 D 中删去.

Step4: 重复 step2、step3 直至 D 为空.

Canopy 聚类因 T_1, T_2 的强相关、弱相关的距离划分,其结果子集允许有重复部分的存在,这增加了算法的容错性以及削弱了消除离群孤立点的影响. 再者,因划分子集的存在,只需在子集中进行聚类即可,避免了对所有点求相似度而带来的计算量的问题. 因此,人们可以运用 Canopy 算法对数据进行“预处理”,之后再对处理后的数据进行下一阶段的聚类分析,并将此过程算法简称为“CTK 算法”.

3 基于 MapReduce 的 CTK 加权聚类改进算法

3.1 改进 Canopy 并行算法

为了解决 Canopy 的区域半径和中心点选取的随意性,本文提出了通过引入近似“梯度值”来确定区域半径以及最优的簇类数.

与传统的聚类算法相同,Canopy 算法同样是将数据集分成不同的簇类 (Canopy), 为了避免局部最优,不同 Canopy 的中心点距离越远越好. 基于以上原则,假设已知 m 个中心点,那么第 $m+1$ 个中心点的选取可以通过以下原则来选取. 可表示为:

$$\text{Dist}_{m+1} = \min \{ d(x_{m+1}, x_r), r = 1, 2, \dots, m \} \quad (1)$$

$$\text{Dist}'_{m+1} = \max \{ \min [d(x_i, x_r)], i \neq 1, 2, \dots, m \} \quad (2)$$

式中, Dist_{m+1} 表示候选中心点与前 m 个中心点最短的距离. Dist'_{m+1} 则表示,最佳的第 $m+1$ 个中心点应该是这些最短距离中的最大值.

通过上述中心点选取策略解决了传统算法随机性. 此外,该选取的策略呈现如下规律: 当最终选取的簇集数大于或小于实际簇集数时, Dist 只呈现微弱的变化. 而当选取的簇集数接近实际时, Dist' 则有较强的浮动. 为了精确簇集的选取,本文引入了梯度值来体现 Dist' 变化的幅度.

$$\text{Gradient}(i) = \frac{|\text{Dist}'(i) - \text{Dist}'(i-1)|}{\text{Dist}'(i-1)} + \frac{|\text{Dist}'(i+1) - \text{Dist}'(i)|}{\text{Dist}'(i)} \quad (3)$$

当 $\text{Gradient}(i)$ 取得大值时,那么数值 i 就达到了 Canopy 算法的最佳簇类数,初始的聚类中心点即为前 i 个中心点,并设置 $\text{Dist}'(i)$ 为区域半径 T_1 .

根据以上理论公式可计算出初始中心集合 Q ,并可通过公式(4)得出区域半径 T_2 .

$$T_2 = \text{Dis}_{\min}(x_i, x_j) \quad i, j \in Q \quad (4)$$

Q 为初始中心点的集合,取初始中心点集合中最近的距离作为区域半径 T_2 .

在分布式环境中,通过最小最大原则来获取初始两点太过耗时,可将初始点的选取简化为距离坐标原点的最大值与最小值点^[13].此外,分布式环境下,为了减少迭代的次数,将算法的迭代次数设置为 $\sqrt{K}$, K 指的是数据集中数据总数.至此,无需人工试凑,即可得到最佳簇集数以及聚类算法的初始状态.

算法 1 找寻初始 Canopy 阈值:

先将数据集 D 归一化预处理,得到数据集 D

Map 阶段:

输入:数据集 D

输出:中心点集合 Q

1)初始状态时,集合 Q 为空.设置迭代次数为 $\sqrt{K}$, K 指的是当前节点处理的数据个数.

2)若集合 Q 为空,计算得到距离坐标原点最近的数据点,将其作为初始点,并放入集合 Q 中.若集合 Q 不为空,根据公式(1)计算得到最佳的下一个中心点,并放入集合 Q 中.

Reduce 阶段:

输入:每个 Map 节点的中心点集合 $Q = \{Q_1, \dots, Q_n\}$

输出:最终中心点集合 U 以及区域半径 T_1, T_2

1)计算集合 Q 的数据个数为 P

2)设置算法的迭代次数为 $\sqrt{P}$,与 Map 阶段类似,根据公式(2)计算得到最佳的中心点,并放入集合 Q 中.

3)计算集合 Q 的数据个数为 L .

4)设置算法的迭代次数为 $\sqrt{L}$,根据公式(3),找到 $\text{Gradient}(i)$ 的最大值,并将集合 Q 中前 i 的中心点放入集合 U ,并设置 $T_1 = \text{Dist}'$.

5)根据公式(5)计算得到区域半径 T_2

改进 Canopy 算法基本思想是,通过引入梯度变化,动态地选择最合适的聚类个数,避免了半径阈值人为选取的盲目性.通过 MapReduce 并行式框架得到中心点集合以及区域半径,并将其作为第二阶段的人参数据.

3.2 基于 MapReduce 改进的 CTK 加权算法的并行化设计

通过改进 Canopy 算法可得初始聚类的中心点以及对应的区域半径 T_1, T_2 ,之后再根据其入参进行数据点归类,并完成聚类迭代.

(1)Canopy 划分

在每个数据节点上加载初始中心点,每个节点计算该节点数据块中的数据点与初始中心点的距离.若距离小于 T_1 ,则加入该中心点.因为一个数据点可能归属于多个中心点,则最终形成相互重叠的 Canopy 簇.

算法 2 Canopy 划分算法:

Map 阶段:

输入:当前数据集 D ,全局中心点 U ,区域半径 T_1

输出:数据点以及所属中心集合

1)While D 不为空

2)从 D 中取出任意数据点 p

3)for $u_i \in U$

4)if $\text{Dist}(p, u_i) < T_1$

5)将 p 点加入到 Center_{u_i}

6)end if

7)end for

8)将 p 点从 D 中删除

9)end while

在划分 Canopy 的过程中,产生了相互重叠的 Canopy 簇,该阶段只有 Map 过程,无 Reduce 函数.最终输出数据点以及所属中心集合 $\langle p, \text{List}(u_i) \rangle$.

2)关键聚类

稳定判定定理:假设有 x_i, c_1, c_2 三个数据点, x_i 是数据空间中任意点, c_1, c_2 为两个聚类中心.假若已知 $2 * d(x_i, c_1) \leq d(c_1, c_2)$,可推出 $(x_i, c_1) \leq d(x_i, c_2)$,即可得到,数据点 x_i 归属于中心点 c_1 ,可将 $2 * d(x_i, c_1) \leq d(c_1, c_2)$ 称为确定性判定.

证明如下:已知 $2 * d(x_i, c_1) \leq d(c_1, c_2)$,将比较符号两侧同时减去 $d(x_i, c_1)$,即 $2 * d(x_i, c_1) - d(x_i, c_1) \leq d(c_1, c_2) - d(x_i, c_1)$,可得 $d(x_i, c_1) \leq d(c_1, c_2) - d(x_i, c_1)$.此外,空间三点满足 $d(x_i, c_1) + d(x_i, c_2) \geq d(c_1, c_2)$,即 $d(c_1, c_2) - d(x_i, c_1) \leq d(x_i, c_2)$.因此,合并两公式可得到 $d(x_i, c_1) \leq d(x_i, c_2)$.表明公式 $2 * d(x_i, c_1) \leq d(c_1, c_2)$ 可推出 $(x_i, c_1) \leq d(x_i, c_2)$,即可得到数据点 x_i 归属于中心点 c_1 .

信息熵加权:首先根据目标簇类构造属性矩阵:

$$\mathbf{X} = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{bmatrix}$$

式中,每一列表示的是一条数据各个维度的属性值,共 n 条数据,数据拥有 m 个维度. 然后计算每个维度在目标簇内的属性比重,用熵值来判断某个指标的离散程度,其熵值越小,该指标对综合评价的影响(即权重)越大,将数据线性变换均一化,使结果映射到 $[0-1]$ 区间. 转换函数如下:

$$M_{ij} = \frac{x_{ij}}{\sum_{i=1}^n x_{ij}} \tag{5}$$

式中, M_{ij} 表示第 i 组数据的第 j 维的比重. 获取数据集中第 j 维属性的信息熵:

$$H_j = -\frac{1}{\ln n} \sum_{i=1}^n M_{ij} \ln M_{ij} \tag{6}$$

转化为权重:

$$\omega_j = \frac{1 - H_j}{\sum_{j=1}^m (1 - H_j)} \tag{7}$$

$$\sum_{j=1}^m \omega_j = 1 \quad j = 1, 2, \dots, m, 0 \leq \omega_j \leq 1 \tag{8}$$

关键聚类是整个算法的核心步骤,其主要算法流程如下:

(1)比较 Canopy 中数据点和对应中心点的距离,根据上述三角判定定理. 若最短的距离的两倍还小于这个聚类中心点到其他中心点的最短距离,则无需额外的比较,直接标记为确定点.

(2)对于确定点进行信息熵加权,确定各属性的权重 ω_j ,将权重的距离公式运用到非确定点,计算其相似性,并归属于最近的中心点.

(3)分别计算不同簇别的数据点均值,进而得到新的聚类中心点,更新全局中心点;

算法在 Map 阶段得到互不重叠的聚类簇集,然进入过 Reduce 阶段计算簇集中数据点的均值,并设置为新的中心点. 之后继续迭代,直至算法收敛. 在迭代过程中既考虑到数据集对于每个维度属性的敏感程度的不同,也考虑到减少相似度比较来缩短算法运算时长.

算法 3 关键聚类:

Map 阶段:

输入:数据点及其所属的中心点集合

输出:中心点以及归属其的数据点

1. while $\langle p, \text{List}(u_i) \rangle$ 不为空

2. for $u_i \in \text{List}(u)$

3. if $\text{Dist}(p, u_i)$ 满足确定性判定

4. 将 p 点加入到 center_{u_i}

5. end if

6. end for

7. 将确定性点集合 center_u 进行信息熵加权计算

8. 将权值 $\sum_{j=1}^m \omega_j$ 加入距离计算,并归属于最近的中心点 u_i

9. 输出 $\langle u_i, p \rangle$

10. end while

此外,在 Map 阶段之后可以通过 Combine 函数在节点内部做“本地 Reduce”,将节点内部相同中心点的数据先行合并,以减少 Reduce 阶段耗时以及网络消耗.

Reduce 阶段:

输入:中心点以及对应的数据点

输出:新的中心点以及区域半径 T_2

1. while $\langle u_i, p \rangle$ 不为空

2. 汇总 map 结果形成 $\langle u_i, \text{List}(p) \rangle$

3. 计算 $\text{List}(p)$ 集合中数据均值作为新的中心点

4. end while

5. if 未达到收敛条件

6. 计算新中心点集合中的 $\text{Dis}_{\min}$ 并作为 T_2

7. 迭代计算

8. end if

3)形成聚类结果.

迭代结束后,只需再进行一次 Map 操作即可,将每个点按照距离归属于各个中心点,形成互补重叠的最终簇集,完成聚类.

4 实验结果与分析

实验在由 Hadoop 集群下完成,由 1 台作为 NameNode,6 台作为 DataNode,其配置均为 2 GB 内存,30 G 硬盘,Hadoop 版本为 2. 8. 0.

(1)单机性能比较试验

试验共分为两组进行,第一组由一台机器执行串型 Canopy 算法,而另一组也采用相同的配置的机器单节点运行本文改进后算法. 在相同数据集的情况下,比较两组的耗时. 其中 t_1 为第一组运行串行算法的单台机器所需要的时间, t_2 为第二组运行本文改进算法的 MapReduce 的耗时. 最终的试验结果如表 1 所示.

表 1 单机串、并行运行时间比较

试验次数	文件大小/KB	t_1 串行时间	t_2 并行时间
1	2 678	39	154
2	5 370	78	234
3	16 100	187	306
4	47 049	478	468
5	50 729	内存溢出	509
6	70 336	内存溢出	619

由表 1 数据可知,在算法的数据集比较小的时候,传统串行算法在处理时间上要小于并行化耗时,这是因为,并行化算法需要承担并行框架所带来的网络传输、资源读取的消耗等。但是,随着数据量的不断增大,串型算法的不足越来越明显,当数据量增大到一定规模的时,系统甚至会出现内存不足的警告。而并行化算法,在数据量不断增加时,它的运行时间优势越加明显。

(2)聚类结果分析

为了验证本文算法的准确性与有效性,针对传统 CTK 算法(Classical CTK)以及本文算法(ICK-KMC)算法在 Iris、Wine 以及 Synthetic Control 数据集上分别进行实验对比,为了对聚类效果进行准确的校验,实验一共进行了 10 次,分别计算每次的正确率以及误差平方和,并将其平均值作为实验结果输出。实验结果如表 2 所示。

表 2 算法准确性、迭代次数比较

数据集	改进前算法			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	0.832	90.27	9	0.908	77.94	7
Wine	0.877	49.96	8	0.923	42.32	6
Synth	0.782	88.58	9	0.813	86.21	8

由表 2 可知,本文算法的正确率高于传统 CTK 算法,此外,在误差平方和以及迭代次数方面也具有较好的性能,说明本文的算法通过引入信息熵加权后,在结果簇集内的数据相似度更高而且算法的收敛速度更快。

为了证明在海量数据集下的算法性能,选取了并行化 K-means 算法(PKC)、传统 CTK 算法(Classical CTK)与本文算法进行比较。分别在上文所述的四个数据集上计算算法的迭代次数以及运行时间。实验结果如图 3、4 所示。

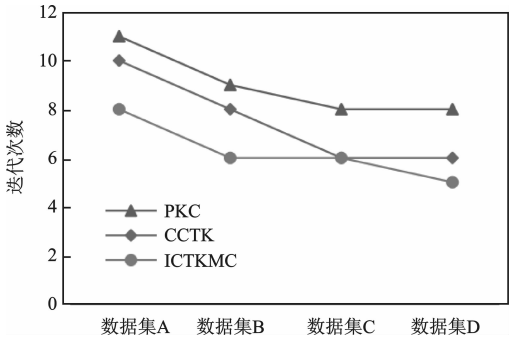


图 3 算法迭代次数对比图

实验结果表明,在迭代次数和运行时间上,传统的

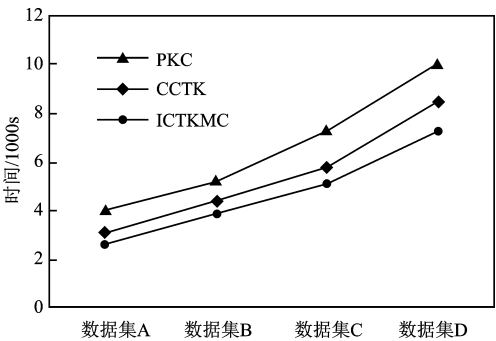


图 4 算法运行时间对比图

CTK 算法和本文算法都引入了区域半径,减少了运算时间,因此明显好于并行的 K-means 算法。在本文算法中,进一步动态改变区域半径,且对求相似度进行了优化,通过进入确定性判定,减少了比较次数,因此更加节省了算法的运行时间,减少了迭代次数。

(3)算法扩展性

为了分析本文算法在 MapReduce 下多节点计算的性能,引入加速比的衡量参数。对于一个计算任务来说,算法的加速比等于单机执行时间除以并行执行时间的值。同样,加速比也是用来衡量算法并行化的重要指标。本文算法加速比曲线如图 5、6 所示。

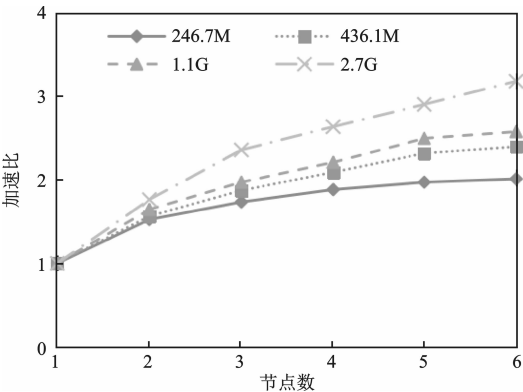


图 5 算法的加速比

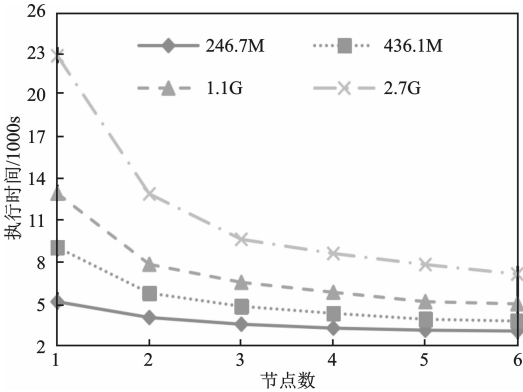


图 6 不同节点算法运行时间对比图

(下转第 114 页)

cracking using probabilistic context-free grammars [C]. Security and Privacy, 2009 30th IEEE Symposium on. IEEE,2009(5): 391-405.

[4] 徐华. 数据挖掘:方法与应用[M]. 北京:清华大学出版社,2014.

[5] 毛国君,段立娟,王实,等. 数据挖掘原理与算法(第二版)[M]. 北京:清华大学出版社,2007.

作者简介:
陈君山 男,(1993-),硕士研究生. 研究方向为网络安全、数据挖掘. E-mail:634073516@qq. com.
刘京菊 女,(1974-),硕士,副教授. 研究方向为网络安全、态势感知.
李振汉 男,(1989-),硕士,讲师. 研究方向为网络安全、数据挖掘.

(上接第 109 页)

由图所示,随着节点数的增加,加速比的增加逐渐缓慢,这是因为节点数的增多同时造成数据的节点网络传输开销的增加. 同时,随着数据量的增加,算法的加速比仍然呈现线性增加的态势,不同大小的数据集均呈现较好的加速比,表明本文算法在大数据集上具有较好的扩展性.

5 结束语

本文以海量数据聚类为背景,改进了传统 Canopy 算法,克服了其随机选择中心点带来的结果不稳定问题,并引入了稳定性判定,增加了信息熵加权,提高了聚类结果的准确性和稳定性,减少了聚类比较次数,同时结合马氏距离度量,并利用 MapReduce 框架设计算法的并行化实现. 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] Hadian A, Shahrivari S. High performance parallel k-means clustering for disk-resident datasets on multi-core CPUs[J]. The Journal of Supercomputing, 2014, 69(2):845-863.

[2] Wu G, Lin H, Fu E, et al. An Improved K-means Algorithm for Document Clustering[J]. Journal of Shandong University, 2008, 43(1):60-64.

[3] Frey B J, Dueck D. Clustering by passing messages between data points[J]. science, 2007, 315(5814):

972-976.

[4] Niknam T, Amiri B. An efficient hybrid approach based on PSO, ACO and k-means for cluster analysis [J]. Applied Soft Computing, 2010, 10(1): 183-197.

[5] Liu J, Han L, Hou L. Cluster Analysis Based on Particle Swarm Optimization Algorithm [J]. Systems Engineering-theory & Practice, 2005(6): 54-58.

[6] Lee G, Tolia N, Ranganathan P. Computing cluster performance simulation using a genetic algorithm solution; U. S. Patent 9,785,472[P]. 2017-10-10.

[7] Elhoseny M, Yuan X, Yu Z, et al. Balancing energy-consumption in heterogeneous wireless sensor networks . using genetic algorithm[J]. IEEE Communications Letters,2015, 19(12): 2194-2197.

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

[9] Shang T, Zhao Z, Guan Z, et al. A DP Canopy K-Means Algorithm for Privacy Preservation of Hadoop Platform[C]//International Symposium on Cyberspace Safety and Security. Springer, Cham, 2017: 189-198.

作者简介:
王万良 男,(1957-),博士,教授. 研究方向为智能调度等.
胡 禹(通讯作者) 男,(1992-),硕士研究生. 研究方向为聚类算法.
E-mail:huyuchina@163. com.