文章编号：1007-5321(2017)增-0068-04 DOI:10.13190/j.jbupt.2017.s.015

改进的 DBSCAN 聚类算法在云任务调度中的应用

王李彧， 孙 斌， 秦 童

（北京邮电大学 信息安全中心，北京 100876）

摘要：针对云计算环境中任务调度中存在的执行效率低的问题，提出了一种基于改进的基于密度的聚类算法(DBSCAN)的云任务调度策略. 首先使用改进的基于密度的聚类算法 DBSCAN 对云任务进行聚类，然后与已经分类的资源进行匹配，解决资源与任务匹配程度低的问题. 实验结果表明，对任务进行聚类后进行任务调度，任务在终端上的平均执行时间减少了大约 35.2%，任务的调度时间也有了明显减少.

关 键 词：任务调度；基于密度的聚类算法；聚类

中图分类号：TP393.1 **文献标志码**：A

Application of Improved DBSCAN Clustering Algorithm in Task Scheduling of Cloud Computing

WANG Li-yu， SUN Bin， QIN Tong

(Information Security Center, Beijing University of Posts and Telecommunications, Beijing 100876, China)

Abstract: Cloud scheduling strategy based on improved density-based spatial clustering of applications with noise (DBSCAN) clustering algorithm was proposed to solve the problem of low efficiency of task scheduling in the implementation of cloud computing environment. Firstly, an improved DBSCAN clustering algorithm was used to cluster tasks. Secondly, the classified tasks were matched with classified resources to solve the low matching degree in resources and tasks. Experiments showed that the average execution time of tasks on the terminal was reduced by about 35.2% after clustering task, and the task scheduling time had also been significantly reduced.

Key words: task scheduling; cloud computing environ ment; cluster

近年来，云计算技术[1-2]成为国内外计算机领域广泛关注的一个研究热点. 它将互联网上闲置的资源整合成一个逻辑上统一的虚拟资源池，最大化利用这些资源为用户提供各类计算或存储任务. 任务调度是在一定规则限制下将任务与资源以优化为目的进行映射，对于任务的执行效率和任务的最终执行效果具有决定性的作用. 基于密度的聚类算法(DBSCAN，density-based spatial clustering of applications with noise)[3]算法是一种基于密度的聚类算法，可以发现任意形状的聚类簇并且聚类结果受噪声影响小. 但该算法中需要 2 个参数：扫描半径(Eps)和最小包含点数(MinPts)，当数据集的密度不均匀时，参数取值不当会影响聚类效果. 对此，提出了一种改进的 DBSCAN 算法，利用粒子群[4](PSO)将数据集划分为 k 个子数据集，获得 k 个不同的 Eps_i，依次调用排序后的 Eps_i 作为参数使用 DBSCAN 算法对样本集 X 进行聚类. 将此聚类方法应用到任务调度中，提高任务与资源的匹配程度，从

收稿日期：2016-05-18

基金项目：国家 242 信息安全计划项目(2015A136)

作者简介：王李彧(1992—)，女，硕士生，E-mail：wangliyu6903@163.com；孙 斌(1967—)，女，副教授.

而改善调度时间与终端任务的执行时间.

1　任务调度介绍

任务调度[5-6]主要目的是提高系统的资源利用率以及服务质量,用户能够获得较高的任务处理速率,服务提供方能获得较高的云计算系统吞吐率.

为了构建按需、廉价的云平台,笔者依据任务在虚拟终端上执行时消耗的 CPU,带宽,内存,磁盘等数据,将任务分类为与资源类别对应的类别[7],例如有些任务用于大文件传输,对网络带宽和磁盘资源要求比较高;持久性链接的网络交互性任务,需要一定的内存来保存链接状态. 鉴于这些情况,将任务分为计算型. 带宽型和磁盘型,在任务调度时根据任务的偏好性寻找对应的虚拟资源,使得各类的资源都能得到充分利用并且提高任务的执行效率.

2　改进的 DBSCAN 算法

2.1　DBSCAN 算法和粒子群算法

DBSCAN 能把高密度的区域划分为一类,并可发现任意形状的聚类. 该算法的具体描述如下.

输入 n 个样本集 $X=\{x_1,x_2,\cdots,x_n\}\in\mathbf{R}^m$, 每个 $x_j=(x_{j1},x_{j2},\cdots,x_{jm})$ 为 m 维向量,两个全局参数:半径 Eps,最少数目 MinPts;

1) 从样本集中抽出一个未处理的样本点,判断输入点是否为核心对象,如果是则找出所有从该点密度可达的对象,将这些点标注为同一簇,否则暂时标注该点为边缘点.

2) 重复步骤 1)直到所有的样本点都被处理.

粒子群算法,初始化为一群随机粒子,每个粒子 i 包含一个 d 维的位置向量 $\boldsymbol{X}_i=(x_{i1},x_{i2},\cdots,x_{\mathrm{id}})^{\mathrm{T}}$ 和速度向量 $\boldsymbol{V}_i=(v_{i1},v_{i2},\cdots,v_{\mathrm{id}})^{\mathrm{T}}$,在每次迭代开始时,粒子根据自己到目前位置发现的最好位置(pBest) $p\mathrm{Best}_i=(p_{i1},p_{i2},\cdots,p_{\mathrm{id}})$ 和目前为止整个群体中所有粒子发现的最好位置(gBest,是 pBest 中的最好值) $g\mathrm{Best}_i=(p_{g1},p_{g2},\cdots,p_{gd})$,利用如下公式来调整自己在第 $k+1$ 次迭代中的第 d 维速度 v_{id}^{k+1} 和自身位置 x_{id}^{k+1}:

$$v_{\mathrm{id}}^{k+1}=\omega v_{\mathrm{id}}^{k}+c_1\mathrm{rand}_1^{k}(p\mathrm{Best}_{\mathrm{id}}^{k}-x_{\mathrm{id}}^{k})+c_2\mathrm{rand}_2^{k}(g\mathrm{Best}_{d}^{k}-x_{\mathrm{id}}^{k}) \quad (1)$$

$$x_{\mathrm{id}}^{k+1}=x_{\mathrm{id}}^{k}+v_{\mathrm{id}}^{k+1} \quad (2)$$

其中:rand_1 和 rand_2 是(0,1)之间的随机数,c_1 和 c_2 被称作学习因子,通常取 $c_1=c_2=2$;ω 是加权系数(惯性权重),取值在 0.1 到 0.9 之间. 粒子通过不断学习更新,最终找到最优解所在位置,即全局最优解 gBest,结束搜索过程. 迭代终止准则为最大迭代次数 $T_{\max}$、计算精度 ε 或最优解的最大停滞步数 Δt.

2.2　算法步骤

提出了 DBSCAN 算法的改进(以下简称 P-DBSCAN),首先利用基于粒子群算法的方法获取较优的初始聚类中心和 k 个不同的 Eps_i;然后分别选取排序后的 Eps_i 进行 DBSCAN 聚类.

关于数据集的划分,涉及如下定义[8].

定义 1　对于一个初始聚类中心,它到离它最近的一个其他初始聚类中心的距离的一半称为该初始聚类中心的划分距离(PartitionDistance).

定义 2　到初始聚类中心的距离不大于 PartitionDistance-ε 的点称为中心点,如果一个区域内的所有点都是某一初始聚类中心的中心点,则称该区域为该初始聚类中心的中心区域(CoreRegin).

定义 3　到初始聚类中心的距离大于 PartitionDistance-ε 且小于 PartitionDistance 的点称为 ε-中心点. 若一个区域内的所有点都是某一初始聚类中心的 ε-中心点,则称该区域为该初始聚类中心的 ε-中心区域(ε-CoreRegin).

定义 4　既不是中心点也不是 ε-中心点的点称为该点的非中心点,若一个区域内的所有点都是某一初始聚类中心的非中心点,则称该区域为非中心区域(Non-CoreRegin).

图 1 所示为点 O 的中心区域、ε-中心区域和非中心区域.

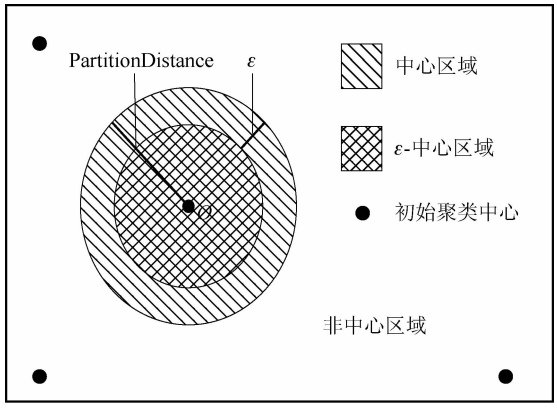

图 1　中心区域、ε-中心区域和非中心区域

P-DBSCAN 的算法流程描述如下.

1) 参数设置:样本数 X,聚类数 K,数据的维数为 d,从样本集 X 中随机抽取 NK 个样本点,初始化粒子的初始搜索点的位置和速度 v_0. 粒子编码结构设计为:$c_{11},c_{12},\cdots,c_{1d},c_{21},c_{22},\cdots,c_{2d},\cdots,c_{k1},c_{k2},\cdots,c_{kd}$,每个粒子由 kd 维数据组成. 其中 c_{ij} $(i=1,2,\cdots,k,j=1,2,\cdots,d)$ 表示第 i 个聚类中心第 j 维.

2) 计算群体中每个个体的适应度值.

3) 如果粒子的适应值大于 pBest,则令 pBest 等于当前位置;若粒子适应度值大于全局最好位置 gBest,则令 gBest 等于目前找到的最好 pBest.

4) 更新每个粒子速度和位置.

5) 若没有达到最大迭代次数 $T_{\max}$ 转入步骤 2) 进行新一轮的迭代,否则输出 gBest 的位置,即最优的 K 个聚类中心.

6) 求每个初始聚类中心的 PartitionDistance.

7) 根据 K 个初始聚类中心的 PartitionDistance 对数据集进行划分.

8) 计算每个数据子集的参数 Eps_i,并对其进行升序排序.

9) 依次调用排序后的 Eps_i 作为参数使用 DBSCAN 算法对样本集 X 进行聚类.

10) 每次调用 DBSCAN 后,对已有聚类的数据点进行标记,标记过的点不再进行后面的 DBSCAN 聚类,直到所有的 Eps_i 使用完毕,没有经过处理的数据点认为是噪声点.

3 基于聚类的任务调度算法

从客户端搜集每个应用在空闲终端上的资源消耗情况,包括 CPU、内存、磁盘、带宽,得到样本集 $X=\{X_1,X_2,\cdots,X_n\}$ 表示 n 个任务的集合,每个任务有 t 个属性,则第 i 个任务的执行情况为 $X_i=\{x_1,x_2,\cdots,x_t\},0\leqslant i\leqslant n$. 使用 P-DBSCAN 算法对样本进行聚类来分析任务对资源的消耗特征.

将 P-DBSCAN 算法应用到任务调度中整个算法的执行过程描述如下.

1) 从客户端搜集各种任务在空闲虚拟终端上的资源消耗情况,做归一化处理后得到 $X=\{X_1,X_2,\cdots,X_n\}$,以此作为 P-DBSCAN 的样本点.

2) 使用 P-DBSCAN 算法对样本集 $X=\{X_1,X_2,\cdots,X_n\}$ 进行聚类.

3) 对步骤 2) 中的聚类结果中的每一类任务对资源的消耗情况进行分析,将任务分为计算型、带宽型和磁盘型.

4) 根据 3) 中聚类的结果,将任务调度给相应的虚拟资源(假定每个资源在资源列表中已有属于自己的标签来标注该资源属于计算型、带宽型或者磁盘型).

4 实验设计与分析

4.1 测试平台介绍

选用笔者团队开发的一个云平台 IVCE 来验证聚类算法应用到任务调度中的效果. 该平台分为应用层、任务层、虚拟资源层和资源管理层. 应用层包括 5 种任务,应用 ID 分别为 App-5,App-18,App-19,App-20,App-21,一个任务的执行情况即为后面聚类时的一个样本点;任务层接受应用层提交上来的任务进行调度并下发到虚拟资源节点;虚拟资源节点执行完任务后将结果返回到 FTP 结果服务器,结果经过收集处理之后将数据插入到数据库中;资源管理层主要管理虚拟资源的注册,资源信息的采集和资源状态的管理.

4.2 实验过程设计

实验中使用以上 5 种应用作为请求任务进行测试,将一种任务发送到终端上执行多次,对收集回来的数据进行处理,从每种任务的数据中选出 200 条可用数据,收集到 1 000 条任务在终端上的消耗情况(CPU、内存、磁盘、带宽),将这 1 000 条数据作为样本集进行聚类得到这 5 种任务的任务类型.

为了验证将聚类算法应用于任务调度中的效果,首先采集每个任务在未改进的任务调度算法中的调度时长和执行时间,然后调度聚类后的任务与计算型、带宽型和磁盘型资源进行匹配后下发并执行,采集此过程中的调度时长和任务执行时间等信息. 最后,将两个过程中的数据进行对比,直观地得到采用聚类算法后云任务调度的效果.

4.3 实验结果与分析

经过对以上 5 种任务在终端上运行对 CPU、带宽、内存、磁盘的消耗情况的结果回收,得到了 1 000 条数据,包括带宽、系统 CPU 占用率、内存占用率及磁盘占用大小,部分数据如表 1 所示,经过聚类之后的结果如表 2 所示.

表 1　聚类样本集中部分样本点

Appid	带宽	CPU/%	内存/%	磁盘/MB
App19	60 168	1.00	9.30	0.55
App19	66 524	1.00	9.30	0.55
App18	3 316	1.20	0.10	2.00
App18	10 905	1.00	0.10	2.00
App20	12 441	1.20	1.60	1.20
App20	1 247	1.10	1.60	1.20
App21	707	1.10	2.60	2.10
App21	2 885	1.00	2.70	2.10
App5	7 349	3.00	1.90	2.70
App5	4 415	7.00	3.40	2.70

表 2　任务聚类结果

类型	应用
带宽型	App-19
计算型	App-20
	App-18
	App-5
磁盘型	App-21

请求 1 000 个任务数，在未使用聚类方法和对聚类后的任务进行调度的情况下，任务在终端上的执行时间如图 2 所示，可以看出，未使用 P-DBSCAN 聚类方法时，任务在终端上的平均执行时间为 5.804 s，使用该方法之后，任务在终端上的平均执行时间减少为 3.761 s，减少了大约 35.2%.

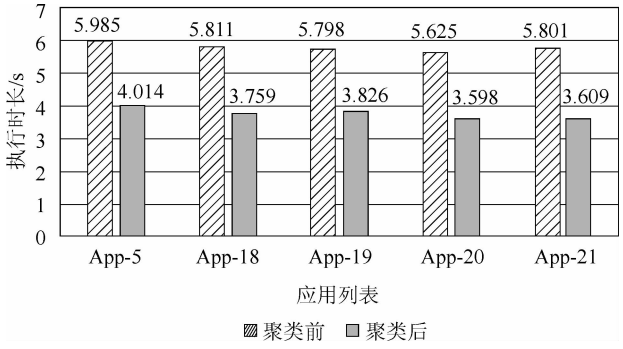

图 2　P-DBSCAN 聚类前后任务执行时长对比

当请求任务数为 1 000、3 000、5 000、7 000、10 000 时，加入聚类算法前后任务调度时长如图 3 所示. 随着请求的任务数量增加，在任务聚类的基础上进行任务调度的优势越来越明显，这是由于当任务数量很大时，加入 P-DBSCAN 对任务进行聚类后，在任务调度时能够很快找到与任务匹配的资源.

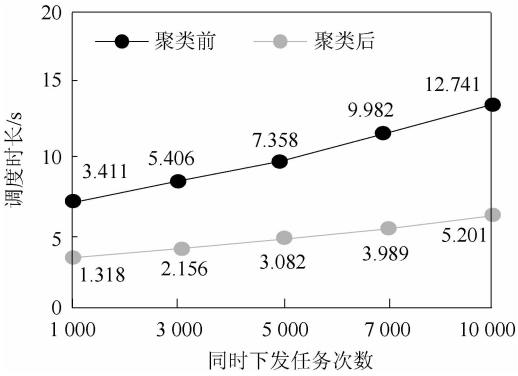

图 3　P-DBSCAN 聚类前后每调度 100 个任务的耗时

5　结束语

首先分析了云计算任务调度中任务与资源的匹配程度问题，进而提出了使用改进的 DBSCAN 算法对任务聚类后与已分类的资源进行匹配. 从实验结果可知，将聚类算法应用到任务与资源的匹配上可以缩短任务的调度时长和任务在虚拟资源上的执行时长，使系统能够尽快满足用户的需求.

参考文献：

[1] Armbrust M, Fox A, Griffith R, et al. A view of cloud computing[J]. Communications of the ACM, 2010, 53(4): 50-58.

[2] Meinel L, Findeisen M, Hes M, et al. Automated real-time surveillance for ambient assisted living using anomni-directional camera[C]// 2004 IEEE International Conference on Consumer Electronics. [S. l.]: IEEE, 2014: 396-399.

[3] Li Xia, Jiang Shengyi, Zhang Qiansheng, et al. A dynamic density-based clustering algorithm appropriate to large - scale text processing [J]. Acta Scientiarum Naturalium Universitatis Pekinensis, 2013, 49(1): 133-139.

[4] 周飞红，廖子贞. 自适应惯性权重的分组并行粒子群优化算法[J]. 计算机工程与应用，2014，50(8)：40-44.

[5] 吴皓. 云环境下任务调度算法研究[D]. 南京：南京邮电大学，2013.

[6] 李丽英. 面向一种云计算平台的任务调度技术研究[D]. 长沙：湖南大学，2011.

[7] 陈廷伟，周山杰，秦明达. 面向云计算的任务分类方法[J]. 计算机应用，2012，32(10)：2719-2723.

[8] 王桂芝，王广亮. 改进的快速 DBSCAN 算法[J]. 计算机应用，2009，29(9)：2505-2508.