面向大规模数据的分层近邻传播聚类算法

刘晓楠[1,2] 尹美娟[1,2] 李明涛[1,2] 姚 东[1,2] 陈武平[3]
(解放军信息工程大学 郑州 450001)[1] (数学工程与先进计算国家重点实验室 郑州 450001)[2]
(信息保障技术重点实验室 北京 100072)[3]

摘 要 近邻传播(Affinity Propagation,AP)聚类具有不需要设定聚类个数、快速准确的优点,但无法适应于大规模数据的应用需求。针对此问题,提出了分层近邻传播聚类算法。首先,将待聚类数据集划分为若干适合AP算法高效执行的子集,分别推举出各个子集的聚类中心;然后对所有子集聚类中心再次执行AP聚类,推举出整个数据集的全局聚类中心;最后根据与这些全局聚类中心的相似度对聚类样本进行划分,从而实现对大规模数据的高效聚类。在真实和模拟数据集上的实验结果均表明,与AP聚类和自适应AP聚类相比,该方法在保证较好聚类效果的同时,极大地降低了聚类的时间消耗。

关键词 数据聚类,近邻传播,分层推举,聚类中心
中图法分类号 TP301.6 **文献标识码** A

Hierarchical Affinity Propagation Clustering for Large-scale Data Set

LIU Xiao-nan[1,2] YIN Mei-juan[1,2] LI Ming-tao[1,2] YAO Dong[1,2] CHEN Wu-ping[3]
(PLA Information Engineering University, Zhengzhou 450001, China)[1]
(State Key Laboratory or Mathematics Engineering and Advanced Computing, Zhengzhou 450001, China)[2]
(Science and Technology on Information Assurance Laboratory, Beijing 100072, China)[3]

Abstract Affinity Propagation (AP) has advantages on efficiency and accuracy, and has no need to set the number of clusters, but is not suitable for large-scale data clustering. Hierarchical Affinity Propagation (HAP) was proposed to overcome this problem. Firstly, the data set was divided into several subsets that can be effectively clustered by AP to select the exemplars of each subset. Then, AP clustering was implemented again on all the subset exemplars to select exemplars of the whole data set. Finally, all the data points were clustered according to similarities with the exemplars, and realizing efficient clustering of large-scale data set. The experimental results on real and simulated data sets show that, compared with traditional AP and adaptive AP, HAP reduces the time consumption greatly and achieves a good clustering result in the meanwhile.

Keywords Data clustering, Affinity propagation, Hierarchical selecting, Clustering center

1 引言

近邻传播(Affinity Propagation, AP)聚类[1]是Frey和Dueck在Science上提出的一种新的聚类算法。该算法通过在节点之间迭代传递吸引度(responsibility)和归属度(availability)这两类消息,推选出聚类样本各自的聚类中心。与以往的聚类方法相比,该算法具有运算速度快和聚类效果好等优势,且不需要预先设定聚类数目,对数据之间的相似度矩阵的对称性没有要求,在人脸识别、基因发现、最优航线确定等方面取得了很好的效果[2-5]。但是该算法仍然存在聚类结果受偏向参数(preference)影响和无法适应大规模数据聚类这两个主要问题[2]。目前针对偏向参数选择问题已有较多解决方法[6-8],而AP算法在大规模数据上的处理效率仍有待提高[9,10]。

本文针对AP算法处理大规模数据聚类问题的局限性,提出一种改进的AP聚类算法——分层近邻传播(Hierarchical Affinity Propagation, HAP)聚类算法。该算法主要分为两步:第一步,通过分层推举获得数据集的聚类中心。即将数据集划分为若干适合AP算法高效执行的子集,在各个子集上分别执行AP聚类推举出各个子集的聚类中心,再对各个子集的聚类中心组成的集合执行AP聚类,从而推举出整个数据集的聚类中心。第二步,对数据集进行全局划分。即以各个聚类中心为初始类,将数据集中的元素划分到与其相似度最大的聚类中心所在的类,从而实现对整个数据集的聚类。

通过分层推举,在各个子集推举出的局部最优聚类中心中再次择优推举,从而推举出整个数据集的最优聚类中心。但是由于待聚类数据仅能从所在子集中选择其局部最优的聚类中心,因此,并不一定能得到最优的聚类结果。而在分层推举出的聚类中心基础上,将每个待聚类数据执行一次全局划分,可以获得更好的聚类结果。HAP聚类算法通过分层推举

到稿日期:2013-05-20 返修日期:2013-10-12 本文受信息保障技术重点实验室开放基金(KJ-12-04)资助。
刘晓楠(1977-),男,博士生,讲师,主要研究方向为高性能计算、软件逆向、数据挖掘,E-mail:97nineday@gmail.com;尹美娟(1977-),女,博士,讲师,主要研究方向为社会网络分析、数据挖掘。

和全局划分，实现了对大规模数据集的高效准确无参数聚类。

本文第 2 节将对 AP 算法进行简要介绍并对其性能进行分析；第 3 节将重点介绍 HAP 算法的具体实现策略；第 4 节将通过实验对提出的算法进行验证。

2 AP 算法及其性能分析

2.1 AP 算法

AP 算法以聚类样本之间的相似度矩阵 $S=(s(i,j))$ 作为输入，输出为样本集的聚类中心以及各样本与聚类中心的归属关系。相似度矩阵中的元素 $s(i,j)$ 表示样本 j 在多大程度上适合作为样本 i 所属类的中心，即类代表点；对角线上的元素 $s(k,k)$ 表示样本 k 的偏向参数，$s(k,k)$ 值越大，样本 k 被选作聚类中心的可能性就越大。

AP 算法引入了两个重要的信息量参数，分别定义为吸引度矩阵 $R(r(i,j))$ 和归属度矩阵 $A(a(i,j))$。AP 算法的迭代过程就是这两个信息量交互更新的过程，两个信息量代表不同的竞争关系，$r(i,j)$ 表示样本 j 适合作为样本 i 的类中心的证据，取决于与其他样本相比，样本 j 作为样本 i 的类中心的最小可能性；$a(i,j)$ 表示样本 i 选择样本 j 作为其类代表点的适合程度，取决于样本 j 作为除样本 i 之外的所有其他样本(包括 j 自身)的类中心的证据之和。

对于任意样本 i，在其他样本中选择对其的吸引度 $r(i,j)$ 和归属度 $a(i,j)$ 之和最大的样本作为其类代表点，即样本 i 的类代表点为 $k=\arg\max_j(a(i,j)+s(i,j))$，类代表相同的样本即聚成了以它们类代表为中心的子类。AP 算法的核心即为 $r(i,j)$ 和 $a(i,j)$ 这两类信息的交替更新，更新公式为：

$$r(i,j)=s(i,j)-\max_{k\neq j}\{a(i,k)+s(i,k)\} \tag{1}$$

$$a(i,j)=\begin{cases}\min(0,r(j,j)+\sum_{k\neq\{i,j\}}\max(0,r(k,j))), & \text{if } i\neq j\\ \sum_{k\neq j}\max(0,r(k,j)), & \text{if } i=j\end{cases} \tag{2}$$

AP 算法在信息更新过程中，为防止发生振荡，引入了阻尼系数(damping factor)$\lambda(\lambda\in[0,1]$，增大 λ 可以降低振荡，λ 的取值一般为 0.9[11])，在每一次迭代过程中，$r(i,j)$ 和 $a(i,j)$ 的更新结果都是由当前迭代过程中的更新值和上一步迭代的结果加权得到的。假设当前迭代次数为 t，则：

$$r^{(t)}(i,j)=(1-\lambda)(s(i,j)-\max_{k\neq j}(a(i,k)+s(i,k)))+\lambda * r^{(t-1)}(i,j) \tag{3}$$

$$a^{(t)}(i,j)=\begin{cases}(1-\lambda)(\min(0,r(j,j)+\sum_{k\neq\{i,j\}}\max(0,r(k,j)))+\\ \lambda * a^{(t-1)}(i,j), & \text{if } i\neq j\\ (1-\lambda)*(\sum_{k\neq j}\max(0,r(k,j)))+\\ \lambda * a^{(t-1)}(i,j), & \text{if } i=j\end{cases} \tag{4}$$

2.2 AP 算法性能分析

AP 算法所能处理的规模有限，主要体现在内存限制和时间消耗两个方面。

2.2.1 处理规模受内存的限制

AP 算法在执行过程中主要的内存消耗是需要存储 3 个矩阵：相似度矩阵 $S(s(i,j))$、吸引度矩阵 $R(r(i,j))$ 和归属度矩阵 $A(a(i,j))$。若矩阵中的元素均为浮点型(4 个字节)，当前的数据处理规模为 N，则存储这 3 个矩阵所需要的内存空间至少为 $3\times N^2\times 4$ 字节。所以在不考虑其他任何内存开销的情况下，AP 算法处理规模与计算机内存容量 M 之间的关系为：$3\times N^2\times 4<M$。以 2G 内存为例，最大处理规模约 1.3 万个数据单位。

2.2.2 时间消耗受数据规模的影响

由式(1)、式(2)可以看出，每一次对 $R(r(i,j))$ 和 $A(a(i,j))$ 进行更新的计算复杂度为 $O(Tn^2)$，其中 T 为迭代次数，因此，随着所处理数据规模的增大，算法的时间消耗也将急剧增大。文献[11]指出，在一台 16G 内存、2.4GHz 主频的计算机上，采用严格的 AP 算法在数小时内能处理 23000 个数据的聚类问题，通过在每次迭代中随机选择节点进行信息交换，对 AP 算法进行改进，能够在一天内处理 120000 个数据的聚类问题。本文在 2G 内存、2.83G 主频的计算机上，以不同规模数据间的相似度矩阵作为输入，采用 AP 算法进行聚类，所需要的平均迭代次数及每次迭代的平均时间如表 1 所列。可以看出，不仅每次迭代的时间消耗呈指数增长，算法收敛所需的迭代次数也呈增长趋势。

表 1 AP 算法对不同规模数据的平均聚类时间

数据规模	平均处理时间(s)	平均迭代次数	每次迭代时间消耗(s)
200	1	14	0.07
400	14	23	0.61
600	49	24	2.04
800	118	24	4.92
1000	280	27	10.37
1200	599	30	19.97
1400	1188	36	33
1600	1964	40	49.1

上述分析及实验表明，在面对一定规模内的聚类时，AP 算法能够高效、准确地执行，但是当数据规模超出一定的范围，AP 算法的效率将降低，甚至无法实现。

3 HAP 聚类算法

本文提出的 HAP 聚类算法将数据集划分为若干个适合 AP 聚类高效执行的子集，通过分层推举和全局划分，实现了对大规模数据集的高效准确无参数聚类。

3.1 相关概念

为便于叙述，对涉及的概念做如下定义：

定义 1(分区) 对待聚类数据集进行划分所形成的用于执行 AP 算法的数据子集。

定义 2(分区代表) 在分区上执行 AP 算法所推举出的该分区的各个聚类中心。

定义 3(底层推举) 在各个分区上执行 AP 算法，获得各分区的分区代表的过程。

定义 4(上层推举) 对各分区代表组成的数据集执行 AP 算法，获得整个数据集的聚类中心的过程。

3.2 算法中的关键问题

在算法实现的过程中，存在以下两个需要解决的关键问题：

3.2.1 分区大小的选择

分区的大小必须在内存所支持的最大处理规模范围内。在考虑内存容量的前提下，分区大小的选择需综合考虑算法

的执行效率和聚类的效果。为获得较好的聚类效果，分区应对数据集具有一定的代表性，其规模不宜过小，分区规模越大对数据集的代表性越好，所获得的聚类效果越好。

在对数据集具有一定代表性的前提下，将总规模为 M 的数据集划分为大小均为 N 的 $K(K=M/N)$ 个分区，则各个分区推举分区代表的时间开销也大致相等（至少在一个数量级），记为 $t(t\propto N^2)$。设各个分区的分区代表占分区大小的比率分别为 $\gamma_i(i=1,\cdots,K)$，则推举出的分区代表总数为 $\sum_{i=1}^{k}\gamma_i \times N=\vec{\gamma}\times M$，记上层推举的时间开销为 $T(T\propto(\vec{\gamma}\times M)^2)$，则算法总的时间开销主要为：$K\times t+T$。在分区对于数据集具有较强代表性的情况下，$\vec{\gamma}$ 与 N 的取值无关，即 T 与 N 的取值无关，而 $K\propto N^{-1}$、$t\propto N^2$，故降低了分区规模 N 能有效提高算法的处理效率。

因此，分区的选择不是越大越好，也不是越小越好，而是应根据实际应用对聚类效果和效率的需求，结合待聚类数据集的特点，选择既能使得分区对数据集具有相当代表性又能很好提高算法效率的分区划分策略。

3.2.2 *偏向参数的选取*

偏向参数 $s(k,k)$ 表示数据点 k 被选作聚类中心的偏向性，对哪些数据点会被选作聚类中心有重要影响。文献[1]建议在没有先验知识的情况下，将所有节点的偏向参数设置为相同值 p；而 p 的大小会对最终的聚类数目产生影响，增大或减小 p 会增加或减少最终的聚类数目。文献[1]建议将 p 设置为相似度矩阵中元素的中位值 pm；文献[12]在采用 AP 算法进行社团发现时，将各个数据点的偏向参数设置为该点与其他数据点的相似度均值，以对不同节点最终被推选为聚类中心的偏向性进行区分，这些偏向参数设置方法并不能保证获得最优的聚类结果。

文献[2,12]提出了自适应的 AP 聚类算法，通过分析 p 的取值空间，在取值空间内动态调整 p 值进行聚类，根据 Silhouette 指标[13,14]对每次聚类结果进行评估，选取其中的最优聚类结果。所不同的是文献[2]根据经验将取值空间设为 $[-\infty,pm/2]$，而文献[12]根据聚类目标能量函数（net similarity）[11]与偏向参数的关系，分别计算聚类个数为 N 和 2 时的偏向参数 $p_{\max}$ 和 $p_{\min}$，将 p 的取值空间设置为 $[p_{\min},p_{\max}]$，从最大值到最小值动态调整偏向参数，每次调整的步进为：

$$p_{step}=\frac{p_{\max}-p_{\min}}{N*0.1*\sqrt{K+50}} \tag{5}$$

其中，$p_{\max}=\max_{i\neq j}s(i,j)$，$p_{\min}=\max_{j}(\sum_{i}s(i,j))-\max_{i\neq j}(\sum_{k}\max(s(i,k),s(j,k)))$，$N$ 为聚类数据集的规模，K 为当前的聚类个数。

Silhouette 指标是对分类、聚类效果进行评估的常用经典指标，假设将有 n 个样本的数据集聚类划分为 k 个子集 $C_i(i=1\sim k)$，$a(t)$ 为 C_j 中的样本 t 与 C_j 中其他样本的相异度均值，$d(t,C_i)$ 为 t 与其他任一子集 C_i 中样本的相异度均值，若 $b(t)=\min_{1\leqslant i\leqslant k}\{d(t,C_i)\}$，则 Silhouette 指标评价样本 t 在此次聚类中类归属的适合度的计算方法为：

$$Sil(t)=\frac{b(t)-a(t)}{\max\{a(t),b(t)\}} \tag{6}$$

整个数据集的聚类效果评价指标为：

$$Sil=\frac{1}{n}\sum_{1\leqslant t\leqslant n}Sil(t) \tag{7}$$

对于每一次聚类结果，获得较大的评价指标则说明聚类效果较好。

自适应的 AP 聚类算法能够得到更好的聚类效果，但需要多次执行 AP 算法，时间消耗较大。正如文献[12]中指出的，以相似度中位值为偏向参数的原始 AP 算法的聚类效果具有较高的准确率，相比于聚类数据的真实情况，其聚类数目相对较多，即聚类的粒度更细。本文采用分层推举的方法，以各个分区推举出的分区代表为基础，通过上层推举对所有分区代表再次聚类，从而推举出整个数据集的聚类中心。因此，在底层推举过程中，不必寻求最优的聚类效果，而是采取粒度较细的聚类，推举出相对广泛的分区代表，这更有助于在上层推举过程中推举出整个数据集中具有更强代表性的聚类中心。因此，本文在底层推举过程中采用各分区样本间的相似度中位值作为偏向参数，而在上层推举过程中采用文献[12]的方法寻求最优聚类结果。

3.3 算法描述

综合以上分析，本文提出的 HAP 聚类算法的流程为：

(1)将数据集划分为 N 个子集 $D=D_1\cup\cdots\cup D_i\cup\cdots\cup C_N$。

(2)对每一个子集 D_i，以 D_i 内样本间相似度中位值作为偏向参数执行 AP 算法，推举出聚类代表集合 E_i。

(3)对所有聚类代表组成的集合 $E=E_1\cup\cdots\cup E_N$ 执行自适应的 AP 聚类，推举出整个数据集的聚类中心 $\{c_1,c_2,\cdots,c_l\}$，$k=l$，将每一个聚类中心 $c_i(1\leqslant i\leqslant k)$ 作为一个初始的类，即 $C_i=\{c_i\}$。

(4)对数据集 D 中每一个样本 d_i，比较其与各个聚类中心 c_j 的相似度 $\mathrm{sim}(d_i,c_j)$，设 $m=\mathrm{argmax}_j\mathrm{sim}(d_i,c_j)$，则 $C_m=C_m\cup\{d_i\}$。

(5)输出聚类结果：$D=C_1\cup C_2\cup\cdots\cup C_k$。算法结束。

4 实验设计及结果分析

本节在 2G 内存、2.83G 主频的计算机上对提出的 HAP、原始 AP 和自适应 AP 这 3 种聚类算法的性能进行实验对比，以说明 HAP 聚类算法在大规模数据聚类上具有更好的适用性。

4.1 实验数据及生成方法

实验采用的数据及其特征（见表 2）中前 5 组：鸢尾花（IRIS）数据、酵母（Yeast）数据、葡萄酒（Wine）数据、汽车估价（Car Evaluation）数据和鲍鱼（Abalone）数据，均为从 UCI 机器学习知识库[15]下载的常用于对聚类算法进行验证的真实数据；后 4 组为计算机生成的模拟数据，因数据属性的维数仅对数据的相似度计算的性能有一定影响，而不会影响聚类算法的性能对比结果，故为简化数据的生成过程，不失一般性地以二维属性对模拟数据进行描述。一个由 s 个类组成的模拟数据集的生成方法是：在一个半径为 R 的圆上选择平均分布的 s 个点作为各类的虚拟中心，将各类的数据样本随机分布在虚拟中心附近半径为 $r(0<r<R\times\sin(0.5\times\frac{2\Pi}{s}))$ 的圆形区域内。各模拟数据集中样本的分布情况如图 1 所示。

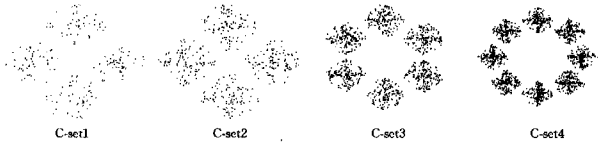

图 1　模拟数据分布图

对上述每组数据，均采用欧氏距离计算组内任意两样本 o_i 和 o_j 的相似度，即：

$$\mathrm{sim}(o_i,o_j)=-\sqrt{\sum_{k=1}^{m}(o_{i_k}-o_{j_k})^2} \tag{8}$$

式中，m 为样本的属性个数，o_{i_k} 和 o_{j_k} 分别为样本 o_i 和第 k 个属性的值。

3 种聚类算法的阻尼系数统一设置为 0.9，HAP 聚类的分区大小选择为数据集规模的 0.2 倍(若该策略的分区大小大于 800，则将分区大小设为 800)。

4.2　聚类效果评估

实验采用常用于对聚类结果进行评估的规格化互信息(Normalized Mutual Information，NMI)指标[16]对各算法的聚类效果进行评估。NMI 指标根据互信息评估聚类算法对数据集的划分结果与标准划分之间的相似性，从而对不同算法的聚类效果进行比较。NMI 指标的计算过程如下。

对数据集的一个划分 P^a，其信息熵为：

$$H(P^a)=-\sum_{i=1}^{k_a}\frac{n_i^a}{n}\log(\frac{n_i^a}{n}) \tag{9}$$

式中，n 为数据集的样本规模，k_a 为划分的子集个数，n_i^a 为第 i 个子集的样本个数。

数据集的两个划分 P^a 和 P^b 之间的互信息为：

$$I(P^a,P^b)=\sum_{i=1}^{k_a}\sum_{j=1}^{k_b}\frac{n_{ij}^{ab}}{n}\log(\frac{\frac{n_{ij}^{ab}}{n}}{\frac{n_i^a}{n}\times\frac{n_j^b}{n}}) \tag{10}$$

式中，n_{ij}^{ab} 表示 P^a 的第 i 个子集与 P^b 的第 j 个子集之间相同样本的个数。

以 NMI 度量数据集的两个划分 P^a 和 P^b 的相似性：

$$NMI(P^a,P^b)=\frac{2\times I(P^a,P^b)}{H(P^a)+H(P^b)} \tag{11}$$

任意两个划分的 NMI 值的范围为[0,1]，NMI 值越大说明两个划分越相似，NMI 值等于 1 说明两个划分完全相同。

4.3　实验结果分析

对 9 组数据集分别执行 AP 聚类、自适应 AP 聚类和 HAP 聚类，聚类结果及性能如表 2 所列。

由表 2 中结果可以看出，AP 聚类算法对数据集的划分粒度较细，在小规模数据集上有最低的时间消耗，但聚类效果一般。自适应 AP 聚类具有最好的聚类效果，但随着数据规模的增大，算法的时间消耗急剧增长，导致该算法不适用于稍大规模数据的聚类。除 Yeast 数据集外，在其他 8 组数据集上，HAP 算法的聚类效果明显优于 AP 聚类，而接近自适应 AP 聚类；但在时间消耗上，该算法不仅远低于自适应 AP 聚类，且在较大规模的数据集上，也远低于 AP 算法的时间消耗，随着数据规模的增大，HAP 聚类在时间消耗上的优势愈发明显。

表 2　不同算法的聚类结果及性能对比

数据集	数据集的特征			AP			自适应 AP			HAP		
	数据规模	实际类数	属性维数	聚类类数	NMI	聚类时间(s)	聚类类数	NMI	聚类时间(s)	聚类类数	NMI	聚类时间(s)
IRIS	150	3	4	27	0.498	1	3	0.758	18	3	0.741	3
Wine	178	3	13	14	0.322	1	3	0.438	64	3	0.419	7
Yeast	1484	10	8	43	0.258	1203	7	0.425	86334	4	0.171	109
Car	1728	4	6	19	0.371	1436	3	0.537	90131	4	0.503	121
Abalone	4898	2	12	8	0.404	9682	2	0.692	231088	2	0.688	187
C-set1	200	4	2	10	0.783	1	4	0.982	69	4	0.930	9
C-set2	400	4	2	18	0.656	13	4	1	1597	4	0.957	12
C-set3	900	6	2	24	0.723	177	6	1	10678	6	1	28
C-set4	1400	8	2	47	0.727	963	8	1	80539	8	0.978	78

HAP 聚类在 Yeast 数据集上效果较差的主要原因是：该数据集的样本分布极不均匀，4 个大类和其余 6 个小类之间的规模差距悬殊，这就导致在推举聚类中心的过程中(特别是在对数据集划分后的分层推举)，小类中样本很难收集到足够的证据以成为聚类中心，因此只能被划分到与其相似度规模较大的其他类中。但将微型类与其相似的大类进行合并，在大多数应用中是允许和可行的，对聚类的整体影响较小。

另外，AP 聚类和自适应 AP 聚类必须以数据集中所有样本间的相似度矩阵为输入，而 HAP 聚类在底层推举时只需要以分区内样本间的相似度矩阵为输入，在上层推举时也仅需要以各分区代表间的相似度矩阵为输入，因此，在存储资源消耗上，HAP 聚类也有明显优势。

综上所述，综合聚类效果、时间和存储资源消耗等因素来看，HAP 聚类算法具有较为明显的优势，更适合于大规模数据的高效准确无参数聚类。

结束语　本文采用分层推举的策略，在大规模数据集上结合分层推举和全局划分的方法，实现了对大规模数据集的高效准确无参数聚类。实验结果表明了该方法的有效性。本文仅对两层推举进行了实现和实验验证，当数据规模大到两层推举也无法高效处理时，可对上层推举过程进行再次分层推举，采用多层推举策略实现推举整个数据集聚类中心的过程。因此，在理论上本文方法不受处理数据的规模限制。

同时，由于受分区划分策略和 AP 算法基于证据推举中心的影响，本文方法无法对大规模数据中的微型类进行有效聚类，寻求合理有效的解决方法将是下一步的工作重点。

参 考 文 献

[1]　Frey B J, Dueck D. Clustering by Passing Messages Between Data Points[J]. Science, 2007, 315(5814): 972-976

[2]　王开军，张军英，李丹，等. 自适应仿射传播聚类[J]. 自动化学报，2007，33(12)：1241-1246

(下转第 192 页)

理器 S-ARM,M-ARM 完成传统的自律循环过程,负责系统整体的状态收集、监控、策略选择和实施,并接受 S-ARM 的请求,分析故障并将解决方案传递给 S-ARM,S-ARM 将方案进一步分配给出现故障的虚拟资源,通过虚拟资源上的 Agent 来实施策略,修复故障。此外,CA 过程主要由 S-ARM 来完成,根据知识库的模板来完成 SLA 的匹配。对于 SLA 的违例则由 VM 上的 Agent 来监控,并定期地向 S-ARM 汇报,S-ARM 与 M-ARM 协商后对违例进行记录与处理。协作过程如图 3 所示。

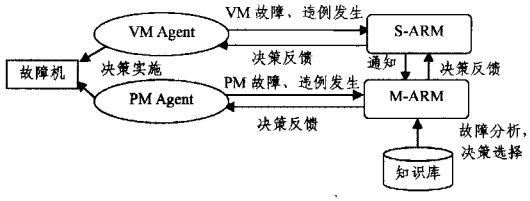

图 3 S-ARM 和 M-ARM 的协作过程

结束语 为了提高云环境下的资源管理效率,解决传统的自律计算模型不能直接应用云环境的问题,本文提出了一个基于分级管理的自律计算模型。针对云环境下的资源虚拟化特征以及需要保证服务的 SLA 等特点,在传统的自律模型中设计了 SLA 匹配的 CA 过程,建立了分级管理的多 Agent 协作体系,使得云环境下的资源管理可以纳入自律循环体系中,从而实现资源管理的高效率。该模型已经应用于某国际合作项目的云系统设计中,实践表明该模型能有效改善云系统的资源管理能力,提高故障修复率并保证云服务的 SLA。

参 考 文 献

[1] Kephart J, Chess D. The Vision of Autonomic Computing [J]. IEEE Computer Society, 2003, 36(1): 41-50

[2] Shi C Y, Zhang W. Agent-Based Computing [M]. Beijing: Tsinghua University Press, 2007: 275-320

[3] Bogdan S, Dan I, Marin L, et al. Towards a real-time reference architecture for autonomic systems[C]//Proceedings-ICSE 2007 Workshops: International Workshop on Software Engineering for Adaptive and Self-Managing Systems. 2007: 1-10

[4] Yu Cheng, Alberto L-G, Source F I. Toward An Autonomic Service Management Framework: A Holistic Vision of SOA, AON, and Autonomic Computing [J]. IEEE Communications Magazine, 2008, 46(5): 138-146

[5] Wang Xiao-ying, Lan Dong-jun, Fang Xing, et al. A resource management framework for multi-tier service delivery in autonomic virtualized environments [C]// Network Operations and Management Symposium, 2008, NOMS 2008. IEEE, 2008: 310-316

[6] Michael M, Ivan B, Vincent C E, et al. Revealing the MAPE loop for the autonomic management of cloud infrastructures [C]// Proceedings-IEEE Symposium on Computers and Communications, ISCC'11. 2011: 147-152

[7] Huebscher, Markus C. A survey of Autonomic Computing-Degrees, models, and applications [J]. ACM Computing Surveys, 2008, 40(3): 7-28

[8] Xu Cheng-zhong. URL: A unified reinforcement learning approach for autonomic cloud management [J]. Journal of Parallel and Distributed Computing, 2012, 72(2): 95-105

[9] Emeakaroha, Vincent C. Towards autonomic detection of SLA violations in Cloud infrastructures [J]. Future Generation Computer Systems, 2012, 28(7): 1017-1029

[10] Champrasert, Paskorn. Exploring self-optimization and self-stabilization properties in bio-inspired autonomic cloud applications [J]. Concurrency Computation Practice and Experience, 2012, 24(9): 1015-1034

[11] You G-W, Hwang S-W, Jain N. Ursa: Scalable load and power management in cloud storage systems [J]. ACM Transactions on Storage, 2013, 9(1): 1-29

(上接第 188 页)

[3] Wang C, Lai J, Suen C, et al. Multi-Exemplar Affinity Propagation[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013, 35(9): 2223-2237

[4] Sakellariou A, Sanoudou D, Spyrou G. Combining multiple hypothesis testing and affinity propagation clustering leads to accurate, robust and sample size independent classification on gene expression data[J]. BMC bioinformatics, 2012, 13(1): 270

[5] Wang L, Zhang L. Color Image Segmentation Algorithm Based on Affinity Propagation Clustering[J]. Foundations of Intelligent Systems. Springer Berlin Heidelberg, 2012, 122: 731-739

[6] 王开军,李健,张军英,等. 半监督的仿射传播聚类[J]. 计算机工程,2007,33(23):197-201

[7] He Yan-cheng, Chen Qing-cai, Xiao-long, et al. An Adaptive Affinity Propagation Document Clustering[C]//Proceedings of the 7th International Conference on Informatics and Systems. Shenzhen, China, 2010: 1-7

[8] Zhong Y, Zheng M, Wu J, et al. Search the Optimal Preference of Affinity Propagation Algorithm[C]//2012 Fifth International Conference on Intelligent Computation Technology and Automation (ICICTA). IEEE, 2012: 304-307

[9] Shang F, Jiao L C, Shi J, et al. Fast affinity propagation clustering: A multilevel approach [J]. Pattern recognition, 2012, 45(1): 474-486

[10] 张震,汪斌强,伊鹏,等. 一种分层组合的半监督近邻传播聚类算法[J]. 电子与信息学报,2013,35(3)

[11] Frey B J. Affinity propagation FAQ [EB/OL]. http://www.psi.toronto.edu/affinitypropagation/faq.html, 2012-01-05/2012-12-01

[12] Ding Fan, Luo Zhi-gang, Shi Jin-long, et al. Overlapping community detection by kernel-based fuzzy affinity propagation[C]// Proceedings of International Workshop on Intelligent Systems and Applications. Changsha, China, 2010: 1-4

[13] Dudoit S, Fridlyand J. A prediction-based resampling method for estimating the number of clusters in a dataset [J]. Genome Biology, 2002, 3(7): 1-21

[14] Rousseuw P J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis [J]. Journal of Computational and Applied Mathematics, 1987, 20: 53-65

[15] Blake C L, Merz C J. UCI repository of machine learning databases[EB/OL]. http://archive.ics.uci.edu/ml/, 2012-05-01/2012-12-01

[16] Fred A L N, Jain A K. Robust Data Clustering[C]//Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Wisconsin, USA, 2003