文章编号:1000-5862(2018)06-0621-06

多目标云资源最佳适应匹配算法的研究

邓 泓[1,2],刘志超[3],王蓝星[3],彭莹琼[1,2*]
(1. 江西农业大学软件学院,江西 南昌 330045;2. 江西省高等学校农业信息技术重点实验室,江西 南昌 330045;
3. 江西农业大学计算机与信息工程学院,江西 南昌 330045)

摘要: 随着云计算的持续研究和发展,面向用户需求的云资源管理与选择是云计算中的重要研究方向之一. 为了满足用户的多种云资源需求,使用树型云资源属性管理表(AMT-Tree)对云资源进行管理,并提出一种多目标云资源最佳适应匹配算法(MoOam),保障用户在海量数据下得到最优资源. 实验结果表明:MoOam 算法在资源匹配中是有效的.

关键词: 云资源匹配;用户需求;树型云资源属性管理表;多目标云资源最佳适应匹配算法
中图分类号: TP 311 **文献标志码:** A **DOI:** 10.16357/j.cnki.issn1000-5862.2018.06.13

0 引言

云计算是近年来迅速发展的一种商业计算模型[1]. 任务调度则是云计算的研究问题之一,它根据用户的各种需求,采用恰当策略把不同的任务分配到云中合适的资源节点上执行[2]. 目前,许多学者进行了大量的研究,如传统网格计算中的调度算法[3-4],以及一些启发式智能算法[5-13],这些算法考虑因素大部分比较单一,并没有从用户各种需求的角度出发. 近年来,不少学者把云平台上的任务调度问题,看作是满足用户需求的资源选择问题. D. Ergu 等[14]根据任务的网络带宽需求、完成时间、计算成本和任务可信度等构建比较矩阵,使用 AHP 计算权重,以此为依据进行资源分配. 该方法满足了多目标的任务需求,但当任务数量较大时,其比较矩阵规模较大,求解速度慢. 刘之家等[15]提出了基于云计算的"用户期待"任务调度算法,该算法综合考虑了用户对云资源的各种固有属性的要求,将其转化为一个期待值,这样在资源选择的过程中可以依照用户的期待函数进行选择. 罗晓峰[16]提出了 QoS 参数映射的方法,建立用户 QoS 需求到云资源的映射模型及云资源的选择、匹配算法,但是选择算法中会遭遇严重的性能问题,对于大量的云资源时处理时间过长.

因此,如何在海量数据下保证多个任务顺利执行,并满足 QoS 满意度、降低计算成本、节约资源等多目标需求的情况下,实现快速地匹配到最优资源是多目标任务调度的关键问题之一. 本文提出一种多目标云资源最佳适应匹配算法(MoOam),保障用户在海量数据下能得到最优资源节点. 本文的 2 个贡献:(i)提出了一种树型云资源管理表(AMT-Tree)结构对云资源进行管理,将相同服务的资源节点进行统一管理,方便搜索;(ii)提出一种多目标云资源最佳适应匹配算法(MoOam),引入灰色关联理论[17],根据数据序列的几何对应关系,在资源池中找到最佳的云资源.

1 问题描述

1.1 云资源匹配模型

云资源的属性有很多种,包括资源本身的能力属性以及 QoS 属性等. 为了应对用户对云资源的各种需求,云服务供应商需部署各种资源来满足云资源消费者的需求. 为了便于描述云资源属性,作如下定义.

定义 1 对每个云资源属性从 0 开始,从小到

收稿日期:2018-02-11
基金项目:国家自然科学基金(71362019),江西省科技计划(20123BBF60177),江西省普通本科高校中青年教师发展计划访问学者专项资金(赣教办函[2016]109)和江西省教育厅科技课题(GJJ14282)资助项目.
作者简介:邓泓(1977-),男,江西都昌人,副教授,主要从事农业信息化与图像处理研究. E-mail:jxaudh@aliyun.com
通信作者:彭莹琼(1978-),女,江西萍乡人,副教授,主要从事计算机视觉处理研究. E-mail:jneyq_pyq@163.com

大依次编号. 设 $\boldsymbol{\alpha}=(a_0,a_1,\cdots,a_n)$,要求 $\forall i(a_{i+1}>a_i)$,$a_i\in\mathbf{Z}$,$i\in\mathbf{Z}$,则称 $\boldsymbol{\alpha}$ 为云资源编号向量. 存储用户需求的云资源编号向量称为云资源目标编号向量. 存储某个资源节点的云资源编号向量称为云资源服务编号向量.

定义 2 存储云资源目标编号向量对应值的向量称为云资源目标向量,存储云资源服务编号向量对应值的向量称为云资源服务向量. 将具有相同编号的一组云资源服务向量存储在一个矩阵中,称这个矩阵为云资源矩阵.

定义 3 设 $\boldsymbol{\alpha}=(a_0,a_1,\cdots,a_n)$ 为云资源目标编号向量,$V_\alpha=(v_0,v_1,\cdots,v_n)$ 是 $\boldsymbol{\alpha}$ 对应的云资源目标向量. $B=\{\boldsymbol{b}_0,\boldsymbol{b}_1,\cdots,\boldsymbol{b}_n\}$ 为云资源服务编号向量的集合,其中 $\boldsymbol{b}_i$ 为云资源服务编号向量;$V_B=\{\boldsymbol{v}_{b_0},\boldsymbol{v}_{b_1},\cdots,\boldsymbol{v}_{b_n}\}$ 是 B 对应的云资源服务向量的集合,其中 $\boldsymbol{v}_{b_i}$ 为云资源服务向量. 并设 $\bar{\boldsymbol{b}}_i=\boldsymbol{b}_i-(\boldsymbol{b}_i-\boldsymbol{\alpha})$,称 $\bar{\boldsymbol{b}}_i$ 为云资源服务编号差向量,$\boldsymbol{v}_{\bar{b}_i}$ 为 $\bar{\boldsymbol{b}}_i$ 对应的云资源服务差向量.

准则 1 若 $\boldsymbol{b}_i$ 能同时满足

$$\boldsymbol{b}_i=\{\boldsymbol{v}_{b_{ij}}\mid\min_i(|\boldsymbol{v}_{b_{ij}}-v_j|)\cap\boldsymbol{\alpha}\subseteq\boldsymbol{b}_i\cap\boldsymbol{v}_{b_{ij}}\in\mathrm{Dom}(a_j)\cap\boldsymbol{v}_{b_{ij}}\in V_B\cap v_j\in V_\alpha\},$$

则 $\boldsymbol{b}_i$ 为最优资源向量. 其中 $\mathrm{Dom}(a_j)$ 是某个目标属性 a_j 的取值集合;$\boldsymbol{v}_{b_{ij}}$ 是某个资源服务向量中的一个属性值.

在准则 1 中,要求最优资源向量的每个属性值与目标向量对应的属性值是最接近的. 然而,当云资源目标编号向量中的资源属性越多,在搜索过程中越难找到最优资源向量. 但可以通过找到与目标向量曲线的几何关系最相近的服务资源向量作为最优资源向量.

准则 2 若 $\boldsymbol{b}_i$ 能同时满足

$$\boldsymbol{b}_i=\{\boldsymbol{v}_{b_{ij}}\mid\max_i(\boldsymbol{V}_\alpha\odot\boldsymbol{v}_{\bar{b}_i})\cap\boldsymbol{\alpha}\subseteq\boldsymbol{b}_i\cap\boldsymbol{v}_{b_{ij}}\in\mathrm{Dom}(a_j)\cap\boldsymbol{v}_{b_{ij}}\in V_B\cap v_j\in V_\alpha\},$$

则 $\boldsymbol{b}_i$ 称为关联度最优资源向量,其中 $\odot$ 为关联度运算. 将关联度最优资源向量也简称为最优资源向量.

本文引入灰色关联理论查找最优资源向量. 灰色关联分析的基本思想是以元素的数据序列为依据,用数学方法研究元素间的几何对应关系. 因此,序列曲线的几何形状越接近,则它们之间的关联度越大.

1.2 灰色关联运算

灰色关联分析的步骤:

(i) 建立数据矩阵;

(ii) 确定参考数据行;

(iii) 对指标数据进行无量纲化:采用均值化法对指标数据进行无量纲化,公式为

$$x_i(k)=\frac{x_i'(k)}{\frac{1}{m}\sum_{k=1}^{m}x_i'(k)};\tag{1}$$

(iv) 计算关联系数:分别计算出云资源矩阵的每个元与目标向量中每个元的关联系数

$$\xi_i=(\min_i\min_k(|x_0(k)-x_i(k)|)+\rho\max_i\cdot\max_k(|x_0(k)-x_i(k)|))/(|x_0(k)-x_i(k)|+\rho\max_i\cdot\max_k(|x_0(k)-x_i(k)|)),\tag{2}$$

其中 ρ 为分辨系数,在(0,1)内取值,若 ρ 越小,关联系数间的差异会越大,区分 QoS 越强. 通常 ρ 取 0.5.

(v) 计算关联度. 根据下式,计算每个云资源向量与目标向量的关联度

$$r_{0i}=\frac{1}{m}\sum_{k=1}^{m}\xi_i(k).\tag{3}$$

1.3 树型属性管理表(AMT-Tree)

为了便于管理各种云资源和更快速地进行云资源匹配,本文建立树型属性管理表对云资源的属性进行统一管理. 树中的每个结点都有一张属性管理表(AMT),属性管理表的数据结构定义如下:AMT(AI,ATA,SAMTA),其中 AI 是每个属性的序号,并按从小到大进行排序. ATA 存储的是某个云资源编号向量对应的云资源矩阵表(AT)的首地址. SAMTA 是子结点属性管理表的地址.

云资源矩阵表(AT)存储的某个云资源编号向量对应的云资源矩阵,其数据结构定义如下:AT(SSV,IP),其中 SSV 是云资源服务向量,IP 是提供此云资源服务向量所在的 IP 地址.

如图 1 所示,树型属性管理表有以下特点:(i)任意一个属性的后续属性管理表的序号都比该属性的序号要大;(ii)在 AMT-Tree 中的每一条从根结点开始的路径代表一个云资源服务编号向量,该路径的终结点的 ATA 表中存储的是此云资源服务编号向量对应的云资源矩阵;(iii)在 AMT-Tree 中的任意一条路径中,相对于父结点中存储的资源,子结点存储中的资源均能满足用户更多需求的资源.

1.4 创建 AMT-Tree

在创建 AMT-Tree 时,云服务供应商应提供云资源配置文件,资源配置文件中为各种资源设置属性,并为属性进行编号,建立云资源编号向量及云资源服务向量. 编号规则由系统统一规定,为了使搜索更加高效,从常用的属性开始编号. 创建 AMT-Tree 算法见表 1.

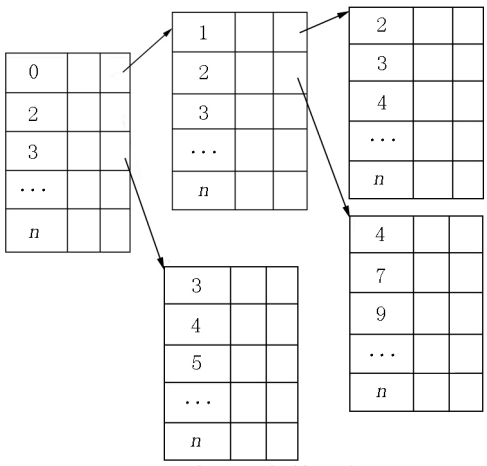

图 1　树型属性管理表

表 1　创建 AMT-Tree 算法

算法 1: 创建 AMT-Tree creatTree( )
输入: 云资源服务编号向量(serverNoV) 、云资源服务向量(serverNoV)
输出: AMT-Tree
初始化: flag = false;
伪代码:
do{
输入 serverNoV 和 serverNoV;
for(遍历 serverNoV) {
if(编号不在对应层的结点中) {
将编号插入到结点相应的位置;
flag = true;
}
}
if(flag = true) {
在路径的终结点上建立资源矩阵;
}
将云资源服务向量插入到资源矩阵中;
} while(是否再输入) ;
return AMT-Tree;

创建 AMT-Tree 算法,是根据云资源提供商提供的云资源服务编号向量来创建树,因此只有云资源服务编号向量才能在树中生成相应的路径,这样在进行资源搜索时能快速确认在 AMT-Tree 中是否存在与目标编号向量一致的路径. 并且,将相同云资源服务编号向量的云资源服务向量放在同一结点中,便于匹配算法时能快速读取服务向量.

根据 AMT-Tree 的特点,资源树创建成功后,资源的增加、退出的操作比较简单,本文的重点是如何搜索到最优资源.

2　MoOam 算法

在 MoOam 算法中采用的是“最佳适应法”,即总是把能满足用户要求,又是资源种类最少的分配给用户. 算法是由创建资源池算法和匹配资源算法组成. MoOam 算法框架见表 2.

表 2　MoOam 算法框架

算法 2: MoOam 算法框架 MoOam(targetNoV,targetV,AMT-Tree)
输入: 云资源目标编号向量(targetNoV) 、云资源目标向量(targetV) 、AMT-Tree
输出: 找到了: 返回最佳匹配资源节点地址 未找到: 返回 Null;
初始化:
伪代码:
if(createPool(targetNoV,AMT-Tree,Pool,index) > 0) {
if(searchSource(Pool, targetV) ! = 0) {
return 最优资源节点地址;
} else{
return false;
}
} else{
return false;
}

2.1　算法框架

MoOam 算法通过云资源目标编号向量搜索云资源服务向量,建立资源池,然后在资源池中匹配出与云资源目标向量关联度最高的资源,即为最优资源.

2.2　创建资源池

创建资源池时,在遍历 AMT-Tree 的过程中,若云资源目标编号向量对应路径的终结点上的资源矩阵能满足要求,则不再搜索其子结点. 创建资源池算法见表 3.

本文通过实验来确定阀值的大小,实验显示阀值与用户要求的属性个数有关,阀值为 n^2+2n. 为了更好保障匹配率,可适当地提高阀值.

2.3　匹配资源

在匹配资源算法中,在进行无量纲化运算时,若发现某个服务向量的属性不符合目标向量的要求,则删除这个向量. 这样既删除了无效向量,也为其后的矩阵运算减少了行数,缩短了运行时间. 匹配资源算法见表 4.

2.4　算法分析

创建 AMT-Tree 树是一个初始化的过程,树的结构是稳定的. 在创建成功后,只需随着资源的增加、退出而做一些简单的操作,所需时间是固定的,本文不作讨论.

表 3 创建资源池算法

算法 3: 创建资源池算法

createPool(targetNoV,AMT-Tree,Pool,index)

输入: 云资源目标编号向量(targetNoV)、AMT-Tree、资源池(Pool)、搜索起点值(index)
输出: 返回资源池中向量的个数
初始化: flag = true; index = targetNoV 中的第 1 个编号
伪代码:

```
if( index > =0) {
if( index 在根结点中) {
    在此结点的子树中搜索云资源目标编号向量对应的路径;
if( 路径存在) {
Pool = Pool + 终结点的资源矩阵;
}
if( Pool 中的向量数 < 阀值) {
    以终结点作为根结点,进行广度遍历; 依次将其后续结点中云资源矩阵中的云资源服务差向量加入到 Pool 中;
} else{
Pool 中的向量数;
        }
    }
    if( Pool 中的向量数 < 阀值) {
        createPool( targetNoV,AMT-Tree,Pool,index - 1);
    } else{
Pool 中的向量数;
    }
}
```

表 4 匹配资源算法

算法 4: 匹配资源算法

searchSource(Pool,targetV)

输入: 资源池(Pool)、云资源目标向量(TargetV)
输出: 若找到,则返回最佳匹配资源节点地址,否则返回 0
初始化: flag = true;
    vertex = V [0];
伪代码:

```
将云资源目标向量插入到资源池矩阵的第 1 行;
for( col = 0…m) {
    for( row = 1…n) {
if( V [row] [col] 不满足 V [0] [col] 的要求) {
    删除 row 行;
    } else{
        vertex [col] = vertex [col] + V [row] [col];
    }
```

表 4(续)

算法 4: 匹配资源算法

searchSource(Pool,targetV)

```
    }
}
if( V 不为空) {
求资源池的矩阵的逆矩阵;
采用(1)式对逆矩阵进行无量纲化;
采用(2)式分别计算出矩阵的每个元与目标向量中每个元的关联系数;
采用(3)式分别计算资源池中每个云资源服务向量与目标向量的关联度;
return 关联度最高的云资源服务向量的地址;
} else {
return 0;
}
```

在 MoOam 算法中,创建资源池算法的最坏情况有 2 种: (i) 用户只需根结点的资源,但只在叶结点才存储了相应资源向量; (ii) 用户只需要根结点的资源,直到搜索至叶结点才能满足阀值要求或者不满足阀值直接结束. 这 2 种情况的搜索时间复杂度都为 $O(2^{n-1})$,其中 n 为树的层数.

在进行资源编号时,将常用的属性优先排在序列的前列,可避免第(i)情况的发生. 只在当云中资源数很少的情况下,才会出现第(ii)情况,因此 MoOam 算法适合在海量资源下搜索资源. 虽然随着用户要求的属性个数增大,阀值也越来越大,但是因算法在搜索到路径的终结点才开始广度遍历,广度遍历只会在一部分子树中进行.

匹配资源算法其实是矩阵运算过程,其时间复杂度为 $O(m \times n)$,其中 m 为服务向量的个数,n 为树的层数. 因在创建资源池时,对 m 值进行了限制,m 值只与包括其服务编号向量的资源数有关,但远小于系统中总的资源数,不会出现因资源总数的增加,运行时间呈指数级增加的情况.

在资源缺乏的情况下,MoOam 算法的时间复杂度为 $O(2^{n-1} + m \times n)$. 但在海量资源下,实际运行时间远低于此,因此,MoOam 算法适合于在海量资源下搜索最佳匹配资源.

3 实验分析

实验分 2 步进行,第 1 步是获取阀值实验,第 2 步实验与文献中的算法[16]进行了性能比对. 实验环境见表 5.

表 5　实验环境

硬件或软件	配置
cpu	I5-3230M 2.6HZ
内存	4G
操作系统	Win7 64 位
编译环境	JDK 8 for Windows, Eclipse 3.7

3.1　确定阀值

算法中阀值的选择决定了算法的运行效率和资源匹配的准确率. 本次实验分别对属性个数从 3 ~ 10 的阀值进行了测试. 每组试验随机生成 2^{n-1} 个不重复的云资源编号向量和 2^{n-1} 个不重复的云资源目标向量(n 为属性的个数) 进行测试. 试验中的每个结点上的云资源服务向量个数初始值为 10,每次以 5 倍增加,直到匹配成功率在 95% 以上结束,此时云资源服务向量的个数即为对应的阀值. 实验数据见表 6.

表 6　阀值表

属性个数	阀值	属性个数	阀值
3	10	7	50
4	15	8	70
5	25	9	95
6	35	10	120

从表 6 的数据可以得出,阀值的计算公式为 n^2+2n.

3.2　算法验证

为了验证本文算法在资源匹配中的有效性,在实验中将其与文献 [6] 中的匹配算法加以比较. 本次实验的运行时间不包括创建 AMT-Tree 树的时间. 因为创建 AMT-Tree 树只是在系统创建初期运行一次,以后在进行资源匹配过程中,不需重复创建.

实验分 3 组进行,每组设定属性的总个数分别为 5、8、12 个. 实验随机产生 200 个的云资源服务编号向量(每个向量的数量为 3 ~ 10 个不等) 和其对应云资源矩阵(每个云资源矩阵含 50 个云资源服务向量) . 在 200 个云资源服务编号向量中随机抽取 50 个作为云资源编号目标向量和其对应的云资源目标向量. 分别使用本方法和文献 [6] 的算法运行结果见表 7.

从表 7 可看出,随着设定属性个数的增加,MoOam 算法运行时间的增加相对恒定,然而文献 [6] 的算法运行时间的增幅比较大. 在相同的匹配率的情况下,MoOam 算法的运行时间更短. 因为 MoOam 算法在树中搜索资源的速度更快,并且在进行关联度计算时已经去掉了许多与目标向量无关的资源,而文献 [6] 的算法的算法要求每次都要遍历所有资源进行计算.

表 7　算法性能对比

算法名	属性个数	运行时间/s	匹配率/%
MoOam 算法	5	17.6	99
	8	22.3	98
	12	38.8	94
文献 [6] 的算法	5	20.2	99
	8	38.7	98
	12	97.5	98

从实验结果可以看出,当属性个数为 12 时,MoOam 算法的匹配成功率并没有文献 [6] 的算法的匹配率高,并且随着属性个数的增加呈下降趋势. 这是因为 MoOam 算法采用灰色关联算法进行资源匹配,要求服务资源的数量达到一定值. 本文在进行实验时,服务资源一直设定为 200,造成 MoOam 算法中某些结点的资源池的数量很少,即进行灰色关联算法时数据矩阵过小,最后导致匹配成功率并没有文献 [6] 的算法的匹配率高.

因此,在面向用户多目标需求的情况下,MoOam 算法在保障用户在海量数据下得到最优资源具有较好的效果.

4　总结与展望

面向用户的各种需求,需要有更加有效的数据结构进行管理和算法选择最优资源. 为了满足用户的多目标需求,本文设计了树型云资源属性管理表(AMT-Tree) 对云资源进行管理,并提出一种多目标云资源最佳适应匹配算法(MoOam) ,引入了灰色关联理论,将与云资源目标向量关系度最高的云资源服务向量分配给用户.

在云资源服务向量极缺的情况下,MoOam 算法并没有优势; 并且 MoOam 算法目前只考虑在海量数据中根据用户多目标需求进行快速匹配最优资源,但将资源的负载均衡考虑其中.

5　参考文献

[1] 刘鹏. 云计算 [M] . 2 版. 北京: 电子工业出版社,2011: 1-3.

[2] 梁庆中. 混合云平台上多目标任务调度算法研究 [D] . 北京: 中国地质大学,2015.

[3] Gaurang Patel, Rutvik Mehta, Upendra Bhoi, et al. Enhanced load balanced min-min algorithm for static meta task scheduling in cloud computing [J] . Procedia Com-

pute Science,2015,57: 545-553.

[4] 史少锋,刘宴兵. 基于动态规划的云计算任务调度研究[J]. 重庆邮电大学学报: 自然科学版,2012,24(6): 687-690.

[5] 熊聪聪,冯龙,陈丽仙,等. 云计算中基于遗传算法的任务调度算法研究[J]. 计算机应用,2016,36(3): 633-636.

[6] 于国龙,催忠伟,左羽. 基于离散粒子群优化的 MPSoC 节能调度算法[J]. 江西师范大学学报: 自然科学版,2016,40(3): 307-311.

[7] Xue Shengjun,Li Mengying,Xu Xiaolong,et al. An Aco-LB algorithm for task scheduling in the cloud environment[J]. Journal of Software,2014,9(2): 466-473.

[8] 王波,张晓磊. 基于粒子群遗传算法的云计算任务调度研究[J]. 计算法工程与应用,2015,51(6): 85-88.

[9] 曹阳,刘亚军,等. 基于遗传-蚁群算法的云计算任务调度优化[J]. 吉林大学学报: 理学版,2016,54(5): 1077-1081.

[10] Guo Lizheng,Zhao Shuguang. Task scheduling optimization in cloud computing based on heuristic algorithm[J]. Journal of Net Works,2012,7(3): 547-553.

[11] 查安民,谭文安. 融合粒子群与蚁群的云计算任务调度算法[J]. 计算机技术与发展,2016,26(8): 24-29.

[12] 张兴国,周东健,李成浩. 基于粒子群蚁群融合算法的移动机器人路径优化规划[J]. 江西师范大学学报: 自然科学版,2014,38(3): 275-277.

[13] 华夏渝,郑骏,胡文心. 基于云计算环境的蚁群优化计算资源分配算法[J]. 华东师范大学学报: 自然科学版,2010(1): 127-134.

[14] Ergu D,Kou Gang,Peng Yi,et al. The analytic hierarchy process: task scheduling and resource allocation in cloud computing environment[J]. Journal of Supercomputing,2013,64(3): 835-848.

[15] 刘之家. 基于云计算的"用户期待"任务调度算法的研究[J]. 大众科技,2011(4): 75-77.

[16] 罗晓峰. 基于 QoS 参数映射的云资源选择与匹配方法研究[D]. 呼和浩特: 内蒙古大学,2015.

[17] 刘思峰,党耀国,方志耕. 灰色系统理论及其应用[M]. 7 版. 北京: 科学出版社,2014.

The Study on Optimal Adaptive Matching Algorithm for Multi-Objective Cloud Resources

DENG Hong[1,2],LIU Zhichao[3], WANG Lanxing[3], PENG Yingqiong[1,2*]

(1. School of Software,Jiangxi Agricultural University,Nanchang Jiangxi 330045,China;
2. Key Laboratory of Agricultural Information Technology of Jiangxi College,Nanchang Jiangxi 330045,China;
3. School of Computer and Information Engineering,Jiangxi Agricultural University,Nanchang Jiangxi 330045,China)

Abstract: With the continuing research and development of cloud computing,the management and selection of cloud resources for user needs is one of the most important aspects of cloud computing. In order to meet the user's multiple cloud resource requirements,the tree of cloud resource attribute management table(AMT-Tree) is proposed to manages the cloud resources,and a multi-objective cloud resource optimal adaptive matching algorithm(MoOam) is proposed,which guarantees the optimal resources of users under massive data. The experiment shows that the MoOam algorithm is effective in resource matching.

Key words: resource matching; user needs; AMT-Tree; MoOam

(责任编辑: 冉小晓)