包簇映射框架下的云资源分配优化算法研究

徐 阳[1] 陈世平[1,2]

[1](上海理工大学光电信息与计算机工程学院 上海 200093)

[2](上海理工大学信息化办公室 上海 200093)

摘 要 在云数据中心,资源分配是云计算系统应用中的核心技术。当前云计算资源在分配过程中,或多或少存在收敛慢、易早熟、资源分配效率低等缺点。为了解决该问题,引入包簇映射框架,提出基于混沌扰动遗传算法。该算法是将遗传算法和混沌搜索机制相结合,以降低成本和提高资源分配效率为目标,利用个体之间的差异性对种群进行初始化,改进种群的交叉变异和适应度函数。通过贪心修正进行合理的云资源分配,采用仿真软件 CloudSim 进行实验,对其性能进行实验分析。实验结果表明,该方法可以有效提高资源分配效率和收敛速度,具有较好的广泛应用价值。

关键词 包簇映射 遗传算法 资源调度 混沌扰动 云计算

中图分类号 TP393.2 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2019.02.007

CLOUD RESOURCE ALLOCATION OPTIMIZATION ALGORITHM BASED ON PACKAGE-CLUSTER MAPPING FRAMEWORK

Xu Yang[1] Chen Shiping[1,2]

[1](*School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China*)

[2](*Network and Information Center Office, University of Shanghai for Science and Technology, Shanghai 200093, China*)

Abstract In cloud data center, resource allocation is the core technology in cloud computing system application. In the current process of cloud computing resource allocation, there are more or less shortcomings such as slow convergence, early maturity and low efficiency. In order to solve this problem, this paper introduced a package-cluster mapping framework and proposed a chaotic disturbance genetic algorithm. This algorithm combined genetic algorithm with chaotic search mechanism, aiming at reducing cost and improving resource allocation efficiency, initialized the population by using individual differences, and improved the crossover mutation and fitness function. We allocated cloud resources reasonably through greedy revision. Through the simulation software CloudSim, we carried out experiments and analyzed its performance. The experimental results show that the method can effectively improve the efficiency of resource allocation and convergence speed, and has good application value.

Keywords Package-cluster mapping Genetic algorithm (GA) Resource scheduling Chaotic disturbance Cloud computing

0 引 言

近年来,随着云计算[1-4]取得了巨大成功,大量高科技公司涌入云服务市场。随着云计算市场规模越来越大,用户群体变得更加复杂,相对应的业务需求也在不断变化。云数据中心资源调度的一个关键目标,是将云数据中心的资源快速充分利用分配给用户。目前

收稿日期:2018-08-26。国家自然科学基金项目(61472256,61170277);上海市一流学科建设项目(S1201YLXK);上海理工大学科技发展基金项目(16KJFZ035,2017KJFZ033);沪江基金项目(A14006)。**徐阳**,硕士,主研领域:计算机网络,云计算。**陈世平**,教授。

主要用的启发式算法有:粒子群算法(PSO)[5]、蚁群算法(ACO)[6]、遗传算法(GA)[7-8]等。这些算法主要目的是减少成本,提高资源利用率,以至于满足用户各方面的要求。但是这些算法都有各自的优缺点,如蚁群算法局部搜索能力较强,但是初始信息素匮乏、初始搜索速度慢等。

当前云资源分配调度方案依然是云计算最关键的技术。为了提高云资源分配效率,减少资源分配的时间,满足用户的使用需求。本文引入包簇理论[9],在包簇映射框架下,提出基于混沌扰动遗传算法 GACD(Genetic Algorithm based on Chaotic Disturbance)。该算法是在遗传算法的基础上引入混沌搜索机制、改进种群个体选择、交叉和变异等操作,提高算法的搜索速度和收敛效率。最后采用 CloudSim 进行实验,以检测算法的正确性。仿真结果表明,基于混沌扰动遗传算法解决了传统遗传算法存在的局部收敛、收敛速度慢和资源分配效率低等缺点,在一定程度上加快了云计算资源分配效率,缩短了任务的完成时间。

1 包簇框架的描述

由文献[9]可知,包和簇是树形结构。最底层的虚拟机组合成一个包,而多个包又组合成一个更高级的包。虚拟机或包所需的资源就是用户对云资源的需求。其中用户对资源需求包括 CPU、RAM、宽带和缓存等。同理,簇是由多个服务器或子簇组成。簇拥有包对资源的需求量。

本文定义有 N 个子簇(或服务器)。p 为子簇标号,$1 \leqslant p \leqslant N$。有 M 个子包(或虚拟机)组成,标号为 v,$1 \leqslant v \leqslant M$。本文的关键是将 M 个子包分配给 N 个子簇,将同一个包中包分配给同一个簇中簇,同一个算法递归调用把子包分配给子簇,直到虚拟机分配给服务器。

由文献[10]可知资源约束:

$$\sum_{v \in V} R_{v,i}[t] X_{V,P}[t] \leqslant A_{p,i}[t] \tag{1}$$

式中:$x_{v,p}[t]$是二进制 0-1 变量:当且仅当在时间 t 时包 v 被分配给簇 p,则 $x_{v,p}[t]=1$;否则,$x_{v,p}[t]=0$。$R_{v,i}[t]$表示包 v 在时间 t 对资源 i 的需求总量。$1 \leqslant i \leqslant J$,$1 \leqslant t \leqslant T$。$A_{p,i}[t]$表示簇 p 在时间 t 时对资源 i 的可用总量。$1 \leqslant p \leqslant N$。

由文献[10]可知簇的运营成本:

$$U(x,y) = \sum_{t=1}^{T} \sum_{p \in \rho} \left(\sum_{v \in V} C_{v,p}[t] x_{v,p}[t] + y_p[t] \hat{c}_p[t] \right)$$

式中:$C_{v,p}[t] = \sum_{i=1}^{J} e_{p,i}[t] R_{v,i}[t]$。$e_{p,i}[t]$表示子簇 p 上对一个单位的资源 i 使用一个时间槽的成本,包括设备折旧、管理开支和能源开支等。$\hat{c}_p[t]$为固定开支,只要簇 p 在时间 t 被使用了,该项固定开支就会发生。$y_p[t]$是一个 0-1 变量,当且仅当存在有包在时间 t 被分配给簇 p 时,它的值为 1。

本文在基于包簇映射的框架下,以减少成本 $U(x,y)$为目标,通过混沌扰动遗传算法进行资源分配,提高分配效率,减少分配时间。

2 基于混沌扰动遗传算法调度策略

2.1 设计思想

基于混沌扰动遗传算法是将遗传算法和混沌搜索机制相结合。首先求出个体之间的差异和适应度值,利用遗传算法进行搜索,混沌机制来避免陷入局部最优,通过精英保留选择法,把优秀的个体作为父代,然后采用改进的交叉和变异操作。算法的主要流程如图 1 所示。

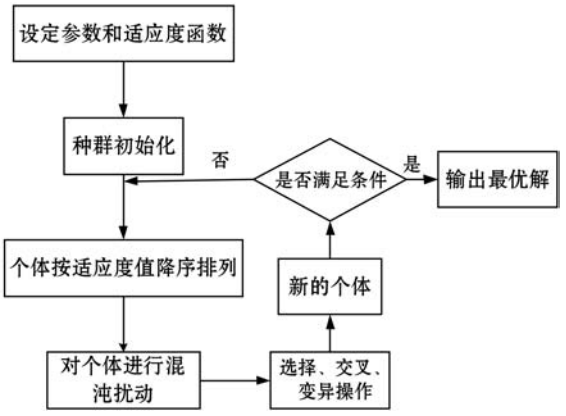

图 1 算法流程图

2.2 个体之间的差异性

种群在初始化时会有很大的机率产生很多相似的个体,导致大量迭代计算,为了解决此问题,本文引出差异性。

差异性是指特征空间中对象之间的差异,将特征空间中不同的对象进行分类。本文用 0-1 编码来对两个个体的基因串进行评估,其中两个对象相同的位置编码相同,则为 0,不同则为 1。0 越多表示两个对象越相同,编码情况越相同。在算法中,限定 0 的个数来避免非常相似的个体。

差异值 $D = \sum_{i=1}^{L} x_{i,1}$,其中 $x_{i,1}$是一个 0-1 变量。当两个对象(包或虚拟机)相同的第 i 位置编码相同时,$x_{i,1}=0$,否则编码 $x_{i,1}=1$。L 为编码长度。

2.3　自适应交叉方式

设种群虚拟机的规模为 M,虚拟机的任务资源需求种类为 J,种群虚拟机的编码长度为 L。第 i 位编码为 0 的个体数为 $S_{i,0}$,为 1 的个体数为 $S_{i,1}$,则定义第 i 位的相似度[12]为:

$$\theta_i = \frac{\max(S_{i,0}, S_{i,1})}{M} \quad i = 1,2,\cdots,L \tag{2}$$

则群体的相似度定义为:

$$\theta = \frac{1}{L}\sum_{i=1}^{L}\theta_i \tag{3}$$

根据 θ 的大小来判断种群虚拟机的收敛程度,θ 越大,表示种群虚拟机越相似,虚拟机所需要的资源也就越相同。种群虚拟机的交叉概率也是依据 θ 的大小动态调整。算法初期,设定较大的交叉概率值,以此加快种群繁衍和收敛。当种群虚拟机的 θ 比较大时,相应地调小交叉概率。参照文献[11]给出相应的自适应调整公式:

$$P_c = t_1 e^{t_2\theta} \tag{4}$$

式中:t_1 和 t_2 的值由 P_c 和 θ 的取值范围共同决定。

2.4　种群初始化和混沌扰动的确定

混沌相关的的内容可以参考文献[12 - 13],混沌的主要思想是利用自身的特点进行搜索,防止陷入局部最优[14 - 15],本文采取 Logistic 映射系统,映射关系如下:

$$\delta_{n+1} = u\delta_n(1-\delta_n) \quad n = 0,1,2,\cdots \tag{5}$$

式中:$u \in [0,4]$,当 $u = 4$ 时为完全混沌状态。

首先,随机产生一个 L 维向量 $\boldsymbol{\delta}_1 = [\delta_{1,1},\delta_{1,2},\delta_{1,l},\cdots,\delta_{1,L}]$,$\boldsymbol{\delta}_{1,l} \in [0,1]$,$\boldsymbol{\delta}_1$ 作为初始向量进行迭代。根据式(5)将得到的各混沌变量映射到优化变量的各维取值空间,然后计算目标函数值,对其值进行排序,从中选取 M 个较好的个体作为初始种群。

为了增加解的多样性,本文用混沌序列对虚拟机种群进行混沌扰动,提高搜索精度,混沌扰动方程如下:

$$\delta'_k = (1-\alpha)\delta^* + \alpha\delta_k \tag{6}$$

式中:δ^* 为最优混沌量,指当前得到的最优解;δ_k 为迭代 k 次后的混沌量;δ'_k 为加了随机扰动后对应的混沌量;α 为扰动系数,$0 < \alpha < 1$。

α 的取值会根据搜索的范围进行调整,在种群初期时,搜索较大范围,α 选取较大的值。到后期时,解已经逐步优化,搜索的范围变小,此时需要选取较小的 α,以便在较小范围内变动。本文按下式确定 α 的值:

$$\alpha = 1 - \left[\frac{k}{k+1}\right]^m \tag{7}$$

式中:m 为参加扰动解的个数;k 为扰动次数,$k = 1,2,\cdots$。

2.5　加速收敛变异法

生物在进化的过程中,一些个体的某些基因位以一定的几率发变异,由原本的 1 变成 0,或者由 0 变成 1,促进了生物群体的进化。然而当两个对象基因相同时,只能依靠基因变异产生新的基因和个体。本文提出一种加速收敛变异法,令个体的基因串前 50% 为高位,后 50% 为低位。在算法初期,首先在全局搜索出适应度高的优秀个体,设置其高位为高变异率,低位为低变异率。在寻找适应度高的优秀个体过程中,高位的变异率逐渐降低至 0,低位的变异率慢慢增加,以此增加解的多样性。

2.6　适应度函数和解的修正

包簇资源分配问题,就是需要求出簇中资源分配到包的最低耗费,使其成本最低。

参照文献[16 - 17],把解的惩罚值引入适应度函数设计中。

惩罚值:该值等于分配到包的总资源与该簇总资源之差的绝对值和二者中较大者的比值。如果被分配到包的总资源与该簇的总资源相差越大,即包需求的总资源越大于或者越小于簇中所拥有的资源,此时惩罚值越大。则构造包簇资源分配的适应度函数如下:

$$fitness(x) = \frac{1}{U(x,y)}\left(1 - \frac{|A_{p,i}[t] - R_{v,i}[t]|}{\max\{R_{v,i}[t], A_{p,i}[t]\}}\right)$$

式中:$U(x,y)$ 为成本函数。

设定成本密度函数 ω 为簇 i 的总成本与总资源的比值:

$$\omega = \frac{U(x,y)}{A_{p,i}[t]}$$

解的贪心修正:首先计算出簇的成本密度值并从小到大进行排序,然后在算出分配给簇的包所需要的总资源,如果超出该簇或服务器所拥有的资源。则需要重新匹配一个簇或服务器。如此循环直到满足包和簇的资源约束为止,得到符合条件的解。

2.7　算法具体步骤

Step 1　设定种群虚拟机的编码和各个参数。

Step 2　种群进行初始化。

Step 3　根据相似度定义,求出每个个体的相似度值,然后按照从小到大排序。

Step 4　选取序列中前 10% 的解,把解转换为十制数,然后除以 $2^n - 1$ 得到小数后,进行混沌扰动,得到 10% 的新个体(且尽量保证个体的适应度值超过上一代),若超过,则转下一步,若未超过则进行迭代,扰

动次数加 1,达到设定值转下一步。

Step 5 对剩余的其他个体进行精英保留选择,自适应交叉和收敛变异操作来产生新的个体。

Step 6 计算出新个体所需资源总量是否超过该簇拥有的资源总量,如果超过了则需要通过贪心修正。

Step 7 首先计算出每个包或者虚拟机的适应度函数值,并将之与上一代中的最大适应度个体进行比较,若新一代适应度函数值较大,则将该值保存、输出转 step 3。否则直接转 step 3。

Step 8 循环 step 3 到 step 7 操作直至进化代数达到设定值,输出最优解。

Step 9 在包最优解的情况下,类似于包的求解,求出包中包的最优资源分配解。一直递归求出最底层虚拟机映射到簇或者服务器的解。

3 仿真实验

本文设计的算法目标有两个,一是降低成本消耗;二是减少资源分配完成时间,提高资源分配效率。为了对算法进行验证,本文在基于包簇映射的框架下设计了 3 组实验,分别是基于混沌扰动遗传算法、简单遗传算法和粒子群算法进行对比实验。

三种算法在 CloudSim 云仿真平台上进行实验。实验环境为 Sun Java SE 7。仿真参数设置:设置种群虚拟机数为 I,每个虚拟机有 J 个任务资源需求。每 q 个虚拟机组合成一个一级包,每 q 个一级包,则生成一个二级包,依次递归生成,直到生成一个最高级的包。

虚拟机的任务资源需求矩阵如下:

$$\boldsymbol{r}_{\text{equires}}=\begin{bmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,J-1} & r_{1,J} \\ r_{2,1} & r_{2,2} & \cdots & r_{2,J-1} & r_{2,J} \\ \vdots & \vdots & & \vdots & \vdots \\ r_{I-1,1} & r_{I-1,2} & \cdots & r_{I-1,J-1} & r_{I-1,J} \\ r_{I,1} & r_{I,2} & \cdots & r_{I,J-1} & r_{I,J} \end{bmatrix}$$

$r_{i,j}$表示虚拟机 i 对资源 j 的需求量。$1\leqslant i\leqslant I, 1\leqslant j\leqslant J$。

最高级包的总资源需求矩阵如下:

$$\boldsymbol{V}_{\text{总}}=\left[\sum_{i=1}^{I} r_{i,1} \quad \sum_{i=1}^{I} r_{i,2} \quad \cdots \quad \sum_{i=1}^{I} r_{i,J-1} \quad \sum_{i=1}^{I} r_{i,J}\right]$$

设虚拟机的二进制编码为 L;二进制编码初始化生成。

设有 H 个服务器,每个服务器有 J 个任务的种类资源。每 p 个服务器组合成一个一级簇,每 p 个一级簇,则生成一个二级簇,依次递归生成,直到生成一个最高级的簇。

服务器资源矩阵如下:

$$\boldsymbol{r}_{\text{esources}}=\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,J-1} & a_{1,J} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,J-1} & a_{2,J} \\ \vdots & \vdots & & \vdots & \vdots \\ a_{H-1,1} & a_{H-1,2} & \cdots & a_{H-1,J-1} & a_{H-1,J} \\ a_{H,1} & a_{H,2} & \cdots & a_{H,J-1} & a_{H,J} \end{bmatrix}$$

$a_{h,j}$表示服务器 h 拥有资源 j 的量。$1\leqslant h\leqslant H, 1\leqslant j\leqslant J$。

最高级簇总资源如下:

$$\boldsymbol{P}_{\text{总}}=\left[\sum_{h=1}^{H} a_{h,1} \quad \sum_{h=1}^{H} a_{h,2} \quad \cdots \quad \sum_{h=1}^{H} a_{h,J-1} \quad \sum_{h=1}^{H} a_{h,J}\right]$$

首先将最高级包先分配给最底层服务器,如果服务器资源不满足包资源的需求,则依次往上分配给一级簇,如果一级簇依旧不满足,则继续递归往上一级簇分配,直到满足包的资源需求为止。这时最高级包的下一级包,开始分配到该簇的下一级簇,依次递归,找到满足需要的最少资源簇即可。类似这种分配,直到虚拟机分配到具体的某个簇或者服务器上即可停止分配。

算法主要是以簇的运营成本 $U(x,y)$ 为目标,分别用基于混沌扰动遗传算法、简单遗传算法和粒子群算法进行实验。实验具体参数如表 1 所示。

表 1 GACD、PSO 和 GA 算法参数设置表

变量	值	说明
H	10	服务器的数量
J	20	资源种类数
L	30	编码长度
Maxgen	500	最大进化代数
D	5	差异值下限
p_c	0.618	交叉概率
p_m	0.03	变异概率
total	50	扰动次数
q	5	包数量或虚拟机数量
p	5	簇数量或服务器数量

小规模数据实验的结果如表 2 所示。

表 2 小规模实验结果表

算 法	虚拟虚数 I	完成时间 t/s
GACD/PSO/GA	10	0.1/0.1/0.1
GACD/PSO/GA	20	0.1/0.1/0.1
GACD/PSO/GA	40	1.1/1.5/1.6
GACD/PSO/GA	60	1.8/2.5/3.3
GACD/PSO/GA	80	3.1/4.3/5.5
GACD/PSO/GA	100	4.6/5.9/6.8
GACD/PSO/GA	120	6.2/8.2/9.8

大规模数据实验的结果如表 3 所示。

表 3　大规模实验结果表

算法	虚拟虚数 I	完成时间 t/s
GACD/PSO/GA	1 000	15.6/18.3/19.6
GACD/PSO/GA	2 000	17.1/25.2/23.4
GACD/PSO/GA	4 000	21.4/45.3/30.7
GACD/PSO/GA	6 000	25.6/56.9/58.0
GACD/PSO/GA	8 000	42.4/90.5/95.2
GACD/PSO/GA	10 000	78.8/200.9/200.2
GACD/PSO/GA	12 000	121.3/301.4/308.3

收敛速度对比如表 4 所示。令虚拟机数量 I = 100,其他参数如表 1 所示。

表 4　算法收敛速度实验对比表

算法	最大进化代数 *Maxgen*	完成时间 t/s
GACD/PSO/GA	50	351/429/431
GACD/PSO/GA	100	250/352/375
GACD/PSO/GA	150	230/301/354
GACD/PSO/GA	200	211/225/268
GACD/PSO/GA	250	201/222/254
GACD/PSO/GA	300	182/218/234
GACD/PSO/GA	350	170/215/231
GACD/PSO/GA	400	165/210/225
GACD/PSO/GA	450	156/199/221

为了实验的准确性,避免偶然情况,本文实验的每个算法都执行 15 次,然后对实验结果取平均值。图 2、图 3 和图 4 是对三种算法的实验结果进行比较。

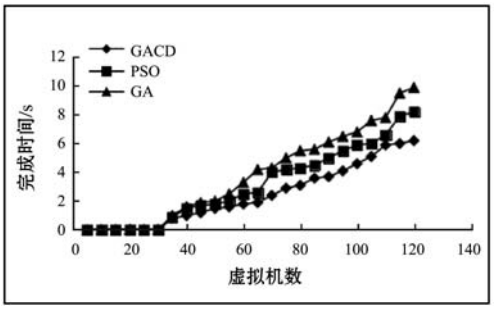

图 2　小规模任务的完成时间比较

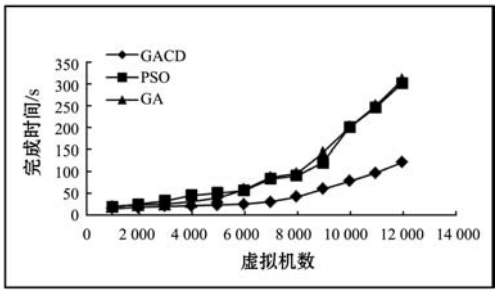

图 3　大规模任务完成时间比较

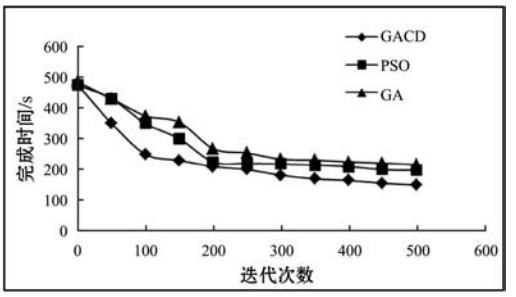

图 4　三种算法的收敛速度对比

从图 2 可以得出,在一定任务范围内,GACD、PSO、GA 算法完成任务的时间差别不大。但从图 3 可以看出,在任务量比较多的情况下,GACD 的任务完成时间最短,优势明显。从图 4 可以看出,本文改进的遗传算法收敛速度明显比较快,算法在迭代 300 次后快速收敛,开始趋向稳定。GACD 算法形成的任务调度方案,对执行包簇映射下的资源分配所需要总时间较少,基本达成了最优化解决方案。

由实验可知,在包簇映射的框架下,基于混沌扰动遗传算法引入混沌扰动机制,利用混沌的遍历性和参数扰动策略,扩大了虚拟机种群的多样性,避免遗传算法陷入局部极小值,增强了遗传算法的全局搜索能力,并且在一定程度上使得该算法具有较高的搜索速度和收敛速度。

在时间上,通过图 2 和图 3 可知,GACD 算法在资源分配的过程中,所需要时间最少,搜索速度最快,PSO 算法次之。这是因为 GACD 算法采用混沌进行种群初始化,利用参数扰动策略进行快速搜索和资源分配。

在收敛速度上,通过图 4 可知,随着进化代数的增加,GACD 算法采用了自适应交叉方式和加速收敛变异法,提高了算法的收敛速度,使得该算法的收敛速度快于 PSO 算法和 GA 算法。

在任务规模上,由图 2 和图 3 可得,GACD 算法效率在小规模任务上略优于 GA 算法和 PSO 算法,但差距不明显,但是随着任务规模的不断扩大,在一定的任务范围之内 GACD 算法明显比 PSO 算法和 GA 算法效率更高。这是因为 PSO 算法和 GA 算法总是尽量把满足条件的资源分配给当前任务,当可用的资源较多时,完成的时间也较快。一旦任务量较大时,此时 PSO 算法和 GA 算法的缺陷也就开始显现出来。

4　结　语

本文主要是在基于包簇映射的框架下,通过改进遗传算法对云资源分配调度进行深入研究,引用混沌扰动机制对遗传算法的搜索操作进行了改进,并且对

选择、交叉和变异操作进一步完善,详细地描述了算法实现步骤和原理。通过实验可以看出,该方法以耗费成本最低为目标,快速找到合适的云资源来进行分配,减少了任务调度的时间,提高了效率,具有较好的效果,实现了较为合理的任务调度方案。

参 考 文 献

[1] Garg S K, Yeo C S, Anandasivam A, et al. Environment-conscious scheduling of HPC applications on distributed Cloud-oriented data centers[J]. Journal of Parallel & Distributed Computing, 2011, 71(6):732-749.

[2] Ergu D, Kou G, Peng Y, et al. The analytic hierarchy process: task scheduling and resource allocation in cloud computing environment[J]. The Journal of Super Computing, 2013,64(3):835-848.

[3] Beloglazov A, Abawajy J, Buyya R. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing[J]. Future Generation Computer Systems, 2012, 28(5):755-768.

[4] Tsong T J, Cen F J, Horng C J. Optimized task scheduling and resourec allocation on cloud computing environment using improved differential evolution algorithm[J]. Computers & Operations Research,2013,40(9):3045-3055.

[5] Zhang L, Chen Y, Sun R, et al. A task scheduling algorithm based on PSO for grid computing[J]. International journal of computational intelligence research, 2008,4(1):37-43.

[6] Xu Z H, Hou X D, Sun J Z. Ant algorithm-based task scheduling in grid computing[C]//Proceedings of the IEEE Canadian conference on electrical and computer engineering, 2003:1107-1110.

[7] 李建锋,彭舰. 云计算环境下基于改进遗传的算法的任务调度算法[J]. 计算机应用,2011, 31(1):184-186.

[8] 齐金平,查显锋. 多任务多资源优化调度的病毒遗传算法[J]. 计算机应用,2011,31(7): 1773-1775.

[9] 卢浩洋,陈世平. 基于包簇映射的云计算资源分配框架[J]. 计算机应用,2016,36(10): 2704-2709.

[10] 王磊,陈世平,卢浩洋. 面向成本优化的包簇机制虚拟化资源分配[J]. 电子科技,2017,30(11):85-88.

[11] 潘立登,黄晓峰. 基于相似度的可变编码长度遗传算法[J]. 北京化工大学学报,1997,24(3):55-59.

[12] Stonebraker M, Brown P, Poliakov A, et al. The architechture of SciDB[C]//Scientific and Statistical Database Management,2011:1-16.

[13] Lu H J, Zhang H M, Ma L H. A new optimization algorithm based on chaos[J]. Zhejiang University Science A,2006,7(4):539-542.

[14] Tavazoei M S, Haeri M. An optimization algorithm based on chaotic behavior and fractal nature[J]. Journal of Computational and Applied Mathematics, 2007, 206(2): 1070-1081.

[15] Choi C, Lee J J. Chaotic local serch algorithm[J]. Artificial Life and Robotics,1998,2(1):41-47.

[16] Runarsson T P, Yao X. Stochastic ranking for constrained evolutionary optimization[J]. IEEE Transactions on Evolutionary Computation, 2000,4(3): 284-294.

[17] Farmani R, Wright J A. Self-adaptive fitness formulation for constrained optimization[J]. IEEE Transactions on Evolutionary Computation, 2003,7(5): 445-445.

(上接第 32 页)

通过设计框架接口的方式,将业务系统的请求转化成为由接口调用的服务,把不同业务系统的异构连接方式转化为标准接口调用的统一服务,由接口实现松耦合的目标。

4 结 语

本文使用标准化建模的方式解决跨域信息共享的数据交换问题。重点描述了信息共享标准化数据模型、信息共享规范和信息交换数据包的构建方式及标准的建模方法。通过接口服务的方式,设计了跨域信息共享服务架构,该架构提供了一种良好的松耦合方式,易于搭建,便于扩展和使用。

参 考 文 献

[1] 戴剑伟. 跨领域信息交换方法与技术[M]. 北京:电子工业出版社,2014.

[2] 王余庆. 面向服务的跨域访问控制系统的设计与实现[D]. 杭州:浙江大学,2010.

[3] 万莹. 区域医疗信息共享协同平台技术研究与应用[D]. 长沙:中南大学,2015.

[4] 仇新红. 一种异构数据交换方法[D]. 长春:吉林大学,2010.

[5] Information Sharing Environment. History[EB/OL]. https://www.ise.gov/about-ise/what-ise/history.

[6] 王少龙. 产品数据共享主模型建模及数据采集技术[D]. 西安:西北工业大学, 2005.

[7] Bouaziz S, Nabli A, Gargouri F. From traditional data warehouse to real time data warehouse[C]//International Conference on Intelligent Systems Design and Applications, 2016: 467-477.

[8] 李鹏. 面向地质勘探的多元异构数据集成关键技术研究[D]. 武汉:中国地质大学,2013.

[9] 于善龙. 基于 XML 的跨应用数据匹配及交换方法的研究与实现[D]. 成都:电子科技大学,2014.