限制解空间的 PSO 云存储任务调度算法*

王 娟, 李 飞, 张路桥
(成都信息工程学院 网络工程学院, 成都 610225)

摘 要: 研究云存储系统中的任务调度算法。分析总结了任务调度在云存储和云计算系统中的不同,指出现有云计算中的 PSO 调度算法应用在云存储中时会产生对云存储系统来说无意义的解,即会要求系统节点提供它所不具有的数据。为解决此问题,改进现有的基于 PSO 的调度算法,引入存在矩阵的概念,将其初始化解和解的更新均限制于存在矩阵中,保证生成的解是有意义解。实验结果显示本调度算法可以节省约 77% 的迭代次数、缩短约 4 倍的执行时间,并保证产生对云存储系统有效的解。

关键词: 云存储; 粒子群算法; 任务调度; 存在矩阵
中图分类号: TP393;TP301.6 **文献标志码**: A **文章编号**: 1001-3695(2013)01-0127-03
doi:10.3969/j.issn.1001-3695.2013.01.031

Task scheduling algorithm in cloud storage system using PSO with limited solution domain

WANG Juan, LI Fei, ZHANG Lu-qiao
(*School of Network Engineering, Chengdu University of Information Technology, Chengdu* 610225, *China*)

Abstract: This paper studied the task scheduling of the cloud storage system. Firstly, it analyzed the differences between the cloud storage and the cloud computing, and pointed out the existing PSO based task scheduling algorithm of cloud computing could not ensure the solution was meaningful to cloud storage, namely some solve may ask nodes offer data that they didn't have. In order to address these problems, the existing PSO based scheduling algorithm was improved by limiting the initialization solution and the solution search space in the exist solution space which could ensure solves were meaningful. The simulation results show that the algorithm can save 77% iteration times and also save about 4 times execution time and offer meaningful solutions for cloud storage system.

Key words: cloud storage; PSO; task scheduling; existing matrix

0 引言

近年来,随着云计算[1,2]技术和软件即服务(SaaS)[3]思想的兴起,云存储成为信息存储领域的一个研究热点。云存储系统作为一种特殊的云系统,提供的是存储服务,即通过网络将本地数据存放在存储服务提供商(SSP)提供的在线存储空间。需要存储服务的用户不再需要建立自己的数据中心,只需向服务提供商申请存储服务,就可以享受异地存储和按需存储的便利,从而避免了存储平台的重复建设,节约了昂贵的软/硬件成本和维护费用[4]。为了安全和效率,同一个数据文件会被冗余存储在不同的多个节点。这些冗余的数据就是常说的副本。在需要这些数据时,用户向云存储系统提交数据请求任务,系统会根据算法选择最优的数据副本和路径给申请者,这就是云存储系统的任务调度。任务调度算法在很大程度上决定了整个系统的效率,是该领域的研究热点之一。

目前,在云计算和网格计算领域已经有较多的任务调度算法被提出[5~8],而现有云存储系统的调度算法多借用这些算法,在任务的编码上略作修改,本质没有区别,专门的云存储系统的研究很少。但是,云存储是一类特殊的云系统,有其区别于传统云计算和网格计算的特点,主要表现在:

a)云计算和网格计算领域的任务是计算型任务,任务可以指派到系统的任意节点执行,区别仅仅在于在高效率(CPU 性能高)节点上执行时间短,而在低效率节点上执行时间长,不会有任务不能由某个节点执行的情况。但是云存储的任务,大多是数据传输任务,首要的前提就是节点有任务所需要的数据,因此某任务不能由某个节点提供数据的情况大量存在,这点导致很多从云计算和网格计算领域移植的算法生成的结果对云存储系统是无效的。

b)云计算和网格计算领域的任务之间存在先后关系,即某些任务必须等到另一些任务的结果后才能执行。而云存储系统的任务一般不存在这种先后关系,数据传输不存在必须先传哪部分后传哪部分。这样使得原有云计算领域中的调度算法可以适当简化,不必考虑任务的先后关系。

以上两点不同提醒人们,现有从传统云计算和网格计算领域移植的算法应该考虑这两大不同,作出改进以适应云存储环境的特点。

本文研究云存储系统的任务调度算法。选择了目前应用较广、评价较好的粒子群算法(PSO)作为基础算法,并根据云

收稿日期: 2012-06-10; **修回日期**: 2012-07-12 **基金项目**: 四川省科技支撑计划项目(2011GZ0195);四川省教育厅科研项目(青年)(10ZB093);成都信息工程学院科研基金资助项目(KYTZ201121);成都信息工程学院青年学科带头人资助项目(J201107)

作者简介: 王娟(1981-),女,四川崇州人,讲师,博士,主要研究方向为网络安全、云计算(wangjuan@cuit.edu.cn);李飞(1966-),男,湖南常德人,教授,硕士,主要研究方向为网格计算和物联网技术等;张路桥(1982-),男,四川自贡人,副教授,硕士,主要研究方向为无线网络、云计算.

存储的特殊要求通过限制其初始解和解空间的方法对现有在云计算环境中应用的 PSO 算法进行改进。实验显示,经过改进的算法能提供原有算法不能提供的对云存储系统的有效解,并大量节省了原有算法的执行时间。

1 云系统任务调度抽象

为了描述方便,先对云系统中的任务调度进行抽象:

本文研究的任务调度是批调度,即在一个设定的单位时间内,会有 n 个新任务产生并等待调度。而不是新产生一个任务,马上进行调度。

a)用 $T=\{t_1, t_2, \cdots, t_n\}$ 表示单位时间等待调度的任务集合,n 是任务数。

b)用 $N=\{n_1, n_2, \cdots, n_m\}$ 表示云系统中的节点集合,假设云系统中有 m 个节点。对于云计算系统来说,n_i 即代表 n_i 上的计算资源;对云存储系统来说,n_i 代表 n_i 上的数据。

c)用 $V=[v_1, v_2, \cdots, v_n]$ 表示任务调度向量,即一个调度方案。对云存储系统来说,v_i 代表第 i 个任务的数据由 v_i 的值代表的资源节点提供,而该向量的长度即是单位时间内要调度任务的总量。例如,某任务调度向量为[4, 1, 3, 1, 2, 6, 5],这个向量的长度是 7,代表该单位时间需要调度的任务数为 7。序号 1 位置上的值是 4,代表任务 1 的数据由系统的 4 号节点提供。以此类推,任务 2 和 4 的数据由节点 1 提供;任务 3 的数据由节点 3 提供;任务 5 的数据由节点 2 提供;任务 6 的数据由节点 6 提供;任务 7 的数据由节点 5 提供。对云计算系统来说,则是代表某个任务被安排到某节点去执行。

d)用 F_v 代表任务调度向量的产生函数。这是调度算法的核心,算法的不同很多就表现在该任务调度向量产生函数的不同上。从某种意义上来说,该产生函数决定了任务调度算法的效率和效果。

e)用 B 代表云系统的带宽矩阵。根据 b)的假设,云系统中有 m 个节点,则 B 是一个 $m\times m$ 的矩阵,每个项 $b(i, j)$ 代表系统节点 i 和 j 之间的带宽。由于节点之间的通路很可能不只一条,则带宽可能有多个取值。现在一般取最小值。

基于以上假设,传统云领域的适应度函数被定义为

$$F=\min\left(\sum_i \frac{t_i}{b_{ij}}\right) \tag{1}$$

其中:i 代表任务序号,j 代表资源节点号。适应度函数实际就是对调度结果好坏的一个评价,它在调度算法中主要有两个作用:(a)调度算法的停止条件之一,即设定一个适应度的阈值,当某一个调度结果的适应度符合(大于或小于,要根据具体适应度定义)阈值则停止算法,对应的调度结果就是该单位时间任务的调度策略,但是,阈值的设定是个难题,目前大多依据经验设置,实际效果差距较大;(b)另一种做法是不设定阈值,转而设定迭代的次数,即由 F_v 产生的调度策略的个数,当达到迭代次数时,算法停止,然后比较各个调度策略的适应度,选择最小/大适应度对应的调度策略为最终结果。在云计算领域中,一般的适应度被定义为"最短总执行时间(minimum makespan)",本文沿用该定义。

整个的云系统调度算法就是要寻找使得总执行时间最小的任务调度向量 V,总执行时间的计算依照式(1)进行,解的产生依照调度策略的生成算法 F_v 的定义。本文取 F_v 为 PSO 算法。下面先介绍云计算中的 PSO 算法,进而指出将其移植到云存储中的不足和改进方法。

2 限制解空间的 PSO 云存储任务调度算法

2.1 PSO 算法在云计算系统任务调度中的应用

1)标准 PSO 算法

在 PSO 中,粒子在解空间中搜索,粒子的位置就代表了一个解,在云系统调度里就代表一个任务调度策略。每个粒子根据局域经验和全局经验更新自己的位置,寻找全局最优解。标准的 PSO 随机初始化一个起始位置,然后在适应度函数的指导下寻找最优解。具体如下:

用 $X_i=(x_{i1}, \cdots, x_{id}, \cdots, x_{iD})$ 代表第 i 个粒子的位置,该粒子的历史最优位置用 $P_i=(p_{i1}, \cdots, p_{id}, \cdots, p_{iD})$ 代表,记为 pbest。全部粒子比较所得的最优位置即全局最优解,记为 gbest。粒子根据速度向量 $V_i=(v_{i1}, \cdots, v_{id}, \cdots, v_{iD})$ 更新自己的位置,更新式如式(2)(3)所示。

$$v_{id}=w\times v_{id}+c_1\times \mathrm{rand}()\times(p_{id}-x_{id})+ c_2\times \mathrm{rand}()\times(p_{gd}-x_{id}) \tag{2}$$

$$x_{id}=x_{id}+v_{id} \tag{3}$$

其中:w 是惯性权重;c_1 和 c_2 是加速度常量也叫学习因子,这里均设为 2,相关研究显示这样设置能使粒子具有自我总结和向群体优秀个体学习的能力;rand()是在[0, 1]之间的随机值。

2)云计算中的 PSO 调度算法

传统基于 PSO 的云计算调度即依照第 1 章步骤 c)的编码方式进行解的编码,然后按照以上算法的描述先随机初始化一个解(随机将任务指派到任意节点执行),判断结束条件,没有结束则根据式(2)(3)产生新的解,再次进行判断,直到达到某一个结束条件。即将任务调度向量的产生函数 F_v 具体化为式(2)和(3)。

将此算法移植到云存储中时,由于解的初始化和更新都没有考虑到云存储的特性,即云存储任务的数据不是任何节点都能提供。因此,该算法常常产生对云存储无效的解。现在一般处理方法是,放弃无效的解重新依据式(2)和(3)产生新解,但是产生的新解也存在无效的问题,反复放弃和产生新解,导致整个算法效率低下。

2.2 限制 PSO 解空间的云存储调度算法

针对云存储环境的特殊要求,本文引入存在矩阵(exist matrix,EM)来指示任务和资源节点间的对应关系。EM 是一个 $n\times m$ 的矩阵,n 是调度任务数目,m 是系统资源节点数目。矩阵的项 e_{ij} 代表任务 i 的数据节点 j 是否有,有则为 1 无则为 0。该矩阵可以从云存储的资源列表中生成。资源列表记录了数据块 ID 和存放资源节点的对应关系,为了保证数据的安全,数据块往往被冗余存储在多个资源节点上,因此存在矩阵中代表任务的一行中会存在多个 1 的情况,对应的列数就是该拥有任务数据的节点 ID。

由于存在矩阵指示的是最小数据块的位置,因此大规模数据量的任务需要被分成多个数据块级的任务进行调度。这点类似于云计算中的海量计算任务被分解成可以并发的小计算任务。不同的是,云计算的计算任务分解后,也不能做到完全的并发,因为有些计算任务必须用到前续任务的计算结果。而云存储不存在必须等到全面数据传输完,才能传输其他任务的

情况,因此云存储任务可以完全并发。这点比云计算任务的分解和调度要简单许多。

在存在矩阵的限制下,对传统云计算中基于 PSO 的任务调度算法作以下两点改进:

a)PSO 的初始化解,不再随机产生,而是根据存在矩阵生成;生成算法简单说来是先从 EM 中获取每行值为 1 的序号,然后从这些序号中随机选择一个作为该行代表任务的解,n 个任务的解构成一个任务调度解 V。

b)在根据式(2)(3)产生新解的时候,先不进行适应度的计算和比较,先进行存在性检测,即产生的新解是否符合存在矩阵中,如果不符合则另外产生一个解直到满足存在矩阵,之后再进入适应性检测。

此外,对于带宽矩阵 B,考虑到一般云存储系统的路由以较大带宽通路为优先选择,在高带宽通路负载较高时才选择低带宽通路,与一般云计算系统中根据目的地地址固定转发地址的处理方式有所不同。因此取最小值就不能比较贴切地描述云存储系统的状况,这里改为取均值作为近似处理。

将云存储系统中的任务调度算法抽象如下所示。

算法 1　存在矩阵限制下的云存储任务调度算法

输入:粒子数目 m,惯性权重 w,加速度常数 c_1 和 c_2,任务向量 T,最大迭代次数 MaxIt,带宽矩阵 B,适应度阈值 f_e,存在矩阵 EM。

输出:任务调度向量 V。

过程:

1. 设置迭代次数 It = 0,从 EM 中生成一个初始解 V_0;

2. while It < = MaxIt

3. 从带宽矩阵 B 中获取带宽 b_{ij};

4. 计算并保存适应度:$f_i = \sum_i \frac{t_i}{b_{ij}}$

5. if f < = f_e then break,输出 V_i;

6. It = It + 1;

7. 依据式(2)(3)更新得到一个新的解 V_{new};

8. 检查 V_{new} 是否符合 EM 的限制,如果不符合则抛弃 V_{new} 重新依照 EM 产生一个 V_{new}

9. end while

选择 f_i 最小的 V_i 输出。

3　实验和分析

3.1　实验环境和实验参数设置

为了便于与现有调度算法作比较,本文采用了已有算法类似的实验网络拓扑,如图 1 所示,包含 10 个节点,编号为 0 ~ 9,存在矩阵由文件分配表实时产生。节点之间连接线上的值代表这两个直接相连点之间的带宽,单位是兆(M)。如 2.2 节所述,间接的两节点之间可能存在多条路径,则定义间接两点之间的带宽为可能路径带宽的平均值。

实验统一设置最大迭代次数为 100 次,10 个任务等待调度。为避免干扰导致的某些个别情况,每个实验重复进行 10 次,每次实验均记录运行时间、迭代次数和适应度值。运行时间就是指算法从开始接受输入需要调度的任务向量开始,到最后输出一个可以接受的调度序列的时间。迭代次数记录的是最后输出的调度序列是第几次迭代产生的,因为最后输出的是最小适应度值(本文指任务总的最短执行时间),这个值可以考察算法的收敛速度,看是否能尽快地找到最小值。适应度值即最后输出的调度序列对应的适应度值,这个值可以考察整个算法陷于局部最优和脱离局部最优的几率。

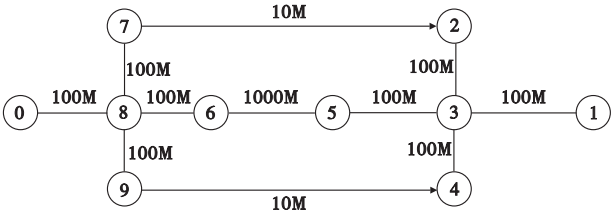

图1　实验网络拓扑

3.2　结果和分析

表 1 和 2 对比展示了传统云计算的 PSO 调度算法和限制解空间的 PSO 调度算法在云存储系统中的运行效果。从运行时间来看,传统云计算 PSO 算法的执行时间均值(4.3 ms)约是本文限制解空间 PSO 算法执行时间均值(1.17 ms)的 4 倍。实际上这个数值受到 MATLAB 时间精度的影响并不是很准确,因为 MATLAB 在运行时只记录到毫秒级,低于 1 ms 的也显示 1 ms。表 2 中多个数据实际执行时间应该是小于 1 ms 的。不过,4 倍的时间差也可以证明本文改进的 PSO 调度算法在云存储环境中具有更高的执行效率。

表 1　云计算 PSO 调度算法应用于云存储中

比较项	实验次数										
	1	2	3	4	5	6	7	8	9	10	mean
运行时间/ms	4	3	3	4	6	4	5	4	5	5	4.3
迭代次数	91	73	59	86	92	93	81	99	88	81	84.3
适应度值	46	157	86	65	46	95	117	90	117	86	90.5

表 2　限制解空间的 PSO 云存储调度算法

比较项	实验次数										
	1	2	3	4	5	6	7	8	9	10	mean
运行时间/ms	1	1	1	1	1.6	1	1	1.7	1.4	1	1.17
迭代次数	11	17	19	17	24	19	17	28	24	13	18.9
适应度值	60	62	87	59	62	67	50	79	91	62	67.9

考察迭代次数,即迭代多少次获得了最优解。传统 PSO 的迭代次数均值为 84.3 次,本文限制解空间的 PSO 算法的迭代次数均值为 18.9 次,仅仅为传统算法的 23%,说明在存在矩阵的限制下算法能迅速找到对云存储系统来说有意义的解。这也是为什么执行时间比传统 PSO 算法短的一个原因。

最后考察生成的调度序列的适应度值。传统算法适应度值为 46 ~ 117,均值 90.5。本文的算法适应度值为 59 ~ 91,均值 67.9。可以看到,从均值上讲,本文算法产生的调度序列的适应度较小,即任务总执行时间较小,这是笔者所希望的。造成较小适应度的原因可能是由于初始解就被指定为一个较小适应度的解,其他解基本围绕它更新而得。而传统云计算 PSO 的初始解是随机产生的,可能产生比较优化的解,也可能产生远离最优解的解。

4　结束语

本文在分析云存储系统固有特征的基础上,指出了云存储系统的任务调度不同于传统的云计算、网格计算的任务调度。从后两者中引入已有的任务调度算法时必须加以改进以符合云存储系统的环境要求。设计了一个可以指明任务和资源节点数据对应关系的矩阵,称为存在矩阵,该矩阵从云存储系统的分布式文件分配表中产生,并将传统 PSO 算法的初始解和解的更新都限制在符合这个存在矩阵的解空间中。实验表明改进后的 PSO 调度算法的执行效率更高,算法 (下转第 154 页)

采用 MATLAB 编写算法程序,参数设置:萤火虫数目 $m=40$,最大吸引度 $\beta_0=1.0$,光强吸收系数 $\gamma=1.0$,步长因子 $\alpha=0.2$,最大迭代次数 max $T=300$。为了验证算法有效性,算法独立运行 100 次。图 2 ~5 中的(a)表示每次迭代的最优值;图 2 ~5 中的(b)表示独立运行 100 次,最优值的分布情况。不同扰动下,FA 对 Car1 问题寻优结果如表 1 所示 。

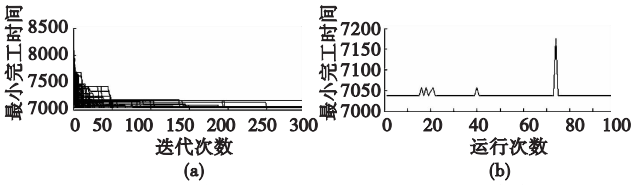

图5 扰动为0.7%时FA对Car1问题寻优结果

表 1 不同扰动下 FA 对 Car1 问题寻优结果

扰动/%	运行次数	最优结果	最优结果击中次数	最好与最坏差值	寻优率/%
0.1	100	7 038	82	172	82
0.3	100	7 038	89	116	89
0.5	100	7 038	94	57	94
0.7	100	7 038	94	137	94

从表 1 可以看出,在不同扰动下,最优结果都是一样的,但最优结果击中的次数是不同的。随着扰动的增加,寻优率基本上是呈增大趋势的,扰动为 0.5% 时,最优结果的波动幅度是最小的且寻优率最大。因此,在 FA 测试 Car1 问题时,当扰动为 0.5% 时,得到最佳情况。

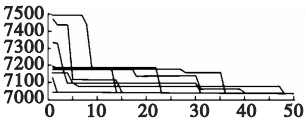

图6 PSO算法对CAR1问题寻优结果

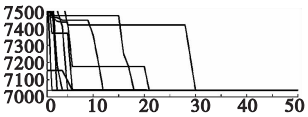

图7 FA对CAR1问题寻优结果

图 6、7 为分别采用基本 PSO 算法和 FA 算法对 CAR1 问题进行仿真测试,两种算法都独立运行了 10 次,每次迭代 50 代。从图中可以得到独立运行 10 次达到最优解 7 038 的迭代次数。从平均来看,PSO 算法在 26 代左右可以达到最优解,FA 在 11 代左右就可以达到最优解。结果表明,虽然都是独立运行 10 次,两种算法都可以全部击中理论最优值(7 038),但 FA 效率明显要高于基本 PSO 算法。

5 结束语

鉴于生产调度的理论和实际意义,研究生产调度的优化算法一直是很多学者研究的热点,并取得了很多成果。本文提出了应用萤火虫算法来求解置换 flow-shop 型生产调度优化问题,通过对经典的 CAR1 问题仿真测试,以及与基本 PSO 算法相比较,得出萤火虫算法求解生产调度问题是可行和有效的。生产调度的研究已有多年,但由于生产调度问题是极其复杂的 NP-hard 问题,因此求解此类问题一直是难点,还有待进一步的研究。国内外对于萤火虫算法应用到生产调度中的研究是很少的,本文对其作了初步的研究,还存在很多问题,还需进行大量的研究工作。

参考文献:

[1] JOHNSON S M. Optimal two and three-stage production schedules with setup times included[J]. Naval Research Logistics Quarterly,1954,1(1):61-68.

[2] 于海斌,薛劲松,王浩波,等. 一种基于神经网络的生产调度方法[J]. 自动化学报,1999,25(4):449-456.

[3] 宋锦和. 基于模拟退火算法的生产调度问题[J]. 长春工程学院学报,2004,5(1):61-63.

[4] 姜桦,李莉,乔非,等. 蚁群算法在生产调度中的应用[J]. 计算机工程,2005,31(5):76-78.

[5] 王万良,唐宁. 微粒群算法的研究现状与展望[J]. 浙江工业大学学报,2007,35(2):136-141.

[6] 曹全军,初红艳,费仁元. 启发式算法和遗传算法在生产调度中的应用[J]. 中国机械工程,2006,17(Z2):211-214.

[7] 刘长平,叶春明. 一种新颖的仿生群智能优化算法:萤火虫算法[J]. 计算机应用研究,2011,28(9):3295-3297.

[8] YANG Xin-she. Nature-inspired metaheuristic algothms[M]. [S. 1.]:Luniver Press,2008:83-96.

[9] 陈子皓,叶春明,寇明顺. 应用新型量子粒子群优化算法求解 PFSP 问题[J]. 技术与创新管理,2012,33(2):162-165.

[10] YANG Xin-she. Firefly algorithms for multimodal optimization[C]//Proc of the 5th International Conference on Stochastic Algorithms: Foundations and Applications. Berlin: Springer-Verlag, 2009: 169-178.

(上接第 129 页)收敛更快,而产生的任务调度序列也具有更小的适应度值,最重要的是调度解完全符合云存储系统需要,不会产生无效解。

下步计划从理论建模上进一步分析限制解空间之后对调度算法的影响,并在此基础上从候选解的产生、筛选等方面进一步改进,使得算法能更快地逼近最优解,获取较小适应度的解。

参考文献:

[1] HAYES B. Cloud computing [J]. Communications of the ACM, 2008,51(7):9-11.

[2] LIN G, DASMALCHI G, ZHU J. Cloud computing and IT as a service: opportunities and challenges[C]//Proc of the 6th IEEE International Conference on Web Services. Los Alamitos: IEEE Computer Society,2008:1-5.

[3] NAMJOSHI J, GUPTE A. Service oriented architecture for cloud based travel reservation software as a service[C]//Proc of the 2009 IEEE International Conference on Cloud Computing. Washington DC: IEEE Computer Society,2009:147-150.

[4] ARMBRUST M, FOX A, GRIFFITH R, *et al*. Above the clouds: a Berkeley view of cloud computing[EB/OL]. 2009. http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html.

[5] TAYAL S. Tasks scheduling optimization for the cloud computing systems[J]. International Journal of Advanced Engineering Sciences and Technologies,2011,5(2): 111-115.

[6] HANG Ruay-shiung, HEN Po-hung. Complete and fragmented replica selection and retrieval in data grids[J]. Future Generation Computer Systems,2007,23(4): 536-546.

[7] RAHMAN R M, ALHAJJ R, BARKER K. Replica selection strategies in data grid[J]. Journal of Parallel and Distributed Computing,2008,68(12):1561-1574.

[8] CHAUHAN S S, JOSHI R C. QoS guided heuristic algorithms for grid task scheduling[J]. International Journal of Computer Applications,2010,2(9):24-31.