浙江理工大学学报，2020，43(4)：527-534
Journal of Zhejiang Sci-Tech University
DOI:10.3969/j.issn.1673－3851(n).2020.04.014

基于改进布谷鸟搜索的Web集群自适应负载均衡算法

张娜[1]，董亮亮[1]，金瑜婷[1]，包晓安[1]，吴彪[2]
(1.浙江理工大学信息学院，杭州 310018；2.日本山口大学东亚研究科，日本山口 753-8514)

摘　要：为了解决Web集群中的高并发访问和资源异构引发的负载不均衡问题，提出了一种基于改进布谷鸟搜索的负载均衡算法。该算法建立了自适应负载分配权重模型，使用基于目标函数的布谷鸟搜索算法寻找最优权重。首先，根据集群任务调度特性定义了负载分配权重，并建立了集群调度的目标函数。其次，将负载分配权重编码为布谷鸟种群个体，使用布谷鸟搜索算法寻找最优个体，结合目标函数进行评估。另外，通过混沌变异增加初始种群的均匀度和离散度，并通过在布谷鸟搜索中引入反向学习，加速了最优权重的输出，根据最优权重将任务调度至集群中的各节点。结果表明：负载均衡算法使得整个集群的服务响应时间降低60%左右，在高并发请求情况下可显著提高异构集群的服务容量和负载均衡度。

关键词：布谷鸟搜索；反向学习；负载均衡；混沌变异；Web集群
中图分类号：TP311.52　　**文献标志码**：A　　**文章编号**：1673-3851 (2020) 07-0527-08

Adaptive load balancing algorithm in web cluster based on improved cuckoo search

ZHANG Na[1], *DONG Liangliang*[1], *JIN Yuting*[1], *BAO Xiaoan*[1], *WU Biao*[2]
(1. School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou 310018, China;
2. Department of East Asian Studies, Yamaguchi University, Yamaguchi 753-8514, Japan)

Abstract: To solve the load-imbalance problem in the web cluster caused by high concurrent access and resource heterogeneity, a load balancing algorithm based on improved cuckoo search was proposed. For this algorithm, an adaptive load distribution weight model was established, and cuckoo search algorithm based on objective function was employed to solve the optimal weights. Firstly, load distribution weights were defined according to the characteristics of cluster task scheduling, and the objective function of cluster scheduling was created. Secondly, load distribution weights were encoded as individuals of cuckoo population, cuckoo search algorithm was applied to figure out the optimal individual, and assessment was made in combination with the objective function. In addition, the uniformity and disperse degree of the initial population were increased by means of chaos mutation, the output of optimal weights was accelerated by introducing opposition-based learning in cuckoo search, and tasks were assigned to nodes in the cluster based on the optimal weights. Experimental results showed that the load balancing algorithm reduces the service response time of the entire cluster by 60% or so, which can significantly increase the service capacity and load balance degree of heterogeneous clusters in the case of high concurrent requests.

Key words: cuckoo search; opposition-based learning; load balancing, chaos mutation; web cluster

收稿日期：2019－11－28　　**网络出版日期**：2020－03－05
基金项目：浙江省重点研发计划项目(2019C03G2060359)；浙江省自然科学基金青年基金项目(Q20F050032)；浙江省公益技术研究计划项目(GG20F010028)
作者简介：张　娜(1977－　)，女，浙江奉化人，副教授，硕士，主要从事软件工程、智能信息处理方面的研究。
通信作者：包晓安，E-mail:baoxiaoan@zstu.edu.cn

0 引 言

随着互联网的高速发展,各类网络化信息系统规模越来越大,“大促”和“秒杀”等业务场景使得访问流量增长迅速[1]。大流量的访问请求远超单个服务器的服务容量,需要Web集群提供大量并发访问服务能力,将集群的总体负载合理地分配到各个节点上,避免单个节点制约集群整体的性能[2]。但是Web集群中各服务器负载能力差异较大,经常出现负载失衡的情况,导致系统利用率低,用户请求得不到快速响应,服务质量急剧下降[3]。因此,如何高效地均衡各服务器节点的负载成为服务器集群需要解决的关键问题。

目前,集群负载均衡问题已取得一定的成果。在经典算法中,加权最少连接调度(Weighted least-connection, WLC)是LVS集群中一种经典的动态负载均衡算法[4],该算法使用服务器节点的连接数作为负载评价标准。Sun等[5]、郑浩等[6]将记录的节点负载作为影响因子来优化任务分配,以实际负载作为依据而不是单一的连接数,这使得量化后的服务器实时状态更有效。随着集群环境的规模化和资源异构化,以负载评估为目标的负载均衡算法已不能满足集群调度的需要,布谷鸟搜索(Cuckoo search, CS)等元启发式算法[7-9]被引入到负载均衡机制中。战非等[10]采用混沌布谷鸟搜索(Chaos cuckoo search, CCS)算法建立了一种云计算资源调度模型,将混沌理论引入布谷鸟算法,有效地提高了资源调度的效率。李佳等[11]提出了基于布谷鸟搜索算法的集群资源调度策略,进一步验证了布谷鸟算法应用于集群负载均衡的可行性与优越性。针对布谷鸟算法本身收敛速度慢的问题,张光亚[12]根据支持向量机(Support vector machine, SVM)建立负载均衡模型,对任务分配进行预测,并使用粒子群-量子的概念改进布谷鸟算法,以此优化SVM最优参数的选择,提高了集群负载预测的精度。赵博颖等[13]提出了一种基于混合并行布谷鸟搜索的作业调度算法,引入两级并行机制,进一步提升了算法的收敛速度。由于布谷鸟搜索的种群个体更新采用随机游走机制,因此以上负载均衡算法存在收敛速度缓慢、易陷入局部最优的问题。

本文针对高并发Web集群环境中的节点负载均衡问题,提出了一种基于改进布谷鸟搜索的Web集群自适应负载均衡算法(Adaptive load balance algorithm in web cluster based on improved cuckoo search, ICS-ALB)。该算法建立了基于目标函数的负载分配模型,自适应地调整负载分配权重,使用改进的布谷鸟搜索算法求解最优负载分配权重,实现Web集群的负载均衡。首先,综合考虑服务器节点的性能和负载指标,建立了负载分配模型和任务调度时间的目标优化方程;然后,考虑负载分配的动态性,根据负载分配权重编码种群个体,并通过混沌变异增加初始化种群的均匀度和连续性;最后,为避免种群寻优后期震荡,引入反向学习对不同阶段的鸟窝位置进行调整,扩大搜索空间,加速最优负载分配权重的输出。

1 模型定义

1.1 Web集群负载分配模型

本文提出的负载均衡算法任务调度模型如图1所示,负载均衡器(Load balancer, LB)将客户端网络请求任务添加到任务队列,收集节点服务器负载指标的实时状态来计算负载分配权重,并采用布谷鸟算法寻找最优权重来进行任务的分配,最终由服务器节点响应客户端请求。假设Web集群由 m 个配置不尽相同的节点服务器,有 n 个未分配的请求任务。负载分配采用先来先服务(First come first serve, FCFS)算法,该算法可以表示为:

$$Q=(T,S,\boldsymbol{W}),$$

其中:$T=\{t_1, t_2, \cdots, t_n\}$ 为待处理的请求任务集合,n 表示任务的个数;集群节点集合 $S=\{s_1, s_2, \cdots, s_m\}$ 表示集群所包含的服务器节点,m 表示节点个数;负载分配权重 $\boldsymbol{W}=(w_1, w_2, \cdots, w_m)$ 表示集群系统任务分配的权重向量。LB根据负载分配权重转发请求 t_i 至节点 s_j,各节点执行结束后直接响应请求。

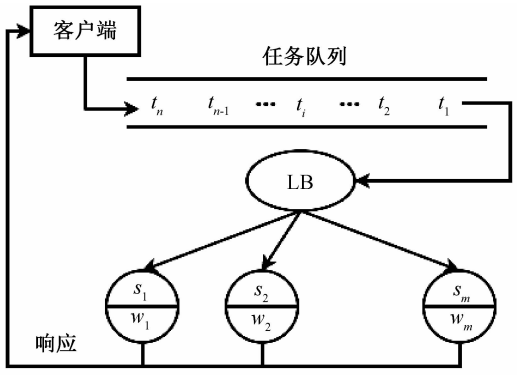

图1 Web集群任务调度模型

1.2 自适应负载分配权重

负载均衡中的各个节点的任务分配权重应与其实时服务容量成正比,选取合适的指标来综合评估

集群中各节点的性能和负载，计算各节点的实时服务容量，从而确定其在整个集群中应分配的任务权重。根据服务器节点程序实际的资源占用情况并参考已有研究中对服务器节点负载的评估[5-6]，本文选取 CPU、内存、磁盘 I/O、网络四个方面来综合评估节点的性能以及负载。

首先，使用 CPU 频率 P_1、内存容量 P_2、磁盘 I/O 速率 P_3、网络带宽 P_4 作为性能指标来评估集群各节点的综合性能，节点 S_i 的综合性能 P 可以用式(1)计算：

$$P(s_i)=\sum_{j=1}^{4}u_jP_j \tag{1}$$

其中：u_j 为各个性能指标的权重参数；P_j 为各项性能指标。节点的各性能指标可认为是常量，u_j 根据集群对节点资源的实际使用情况进行设定。

其次，选择节点的 CPU 利用率 L_1、内存利用率 L_2、磁盘 I/O 占用率 L_3、网络带宽占用率 L_4 作为综合负载率评估指标，节点 S_i 的综合负载率 L 可以用式(2)计算：

$$L(s_i)=\sum_{j=1}^{4}v_jL_j \tag{2}$$

其中：v_j 为各负载指标的权重参数；L_j 为各负载率指标。v_j 体现了各个指标对节点综合负载率的影响程度，需要在任务调度中根据节点实时状态进行调整。当某一项负载指标 L_j 明显高于其他指标时，应增大其权重参数 v_j，以确保节点 s_j 不会因为 L_j 过载而发生故障。

假设 $L_{\max j}$ 为指标 L_j 的过载临界值，由节点本身服务运行所需资源可知 $L_{\max j}$ 取值空间为(0，1)，可根据节点实际资源配置进行设定。使用正向负载指标来实时调整权重参数 v_j，正向负载指标可以用式(3)计算：

$$L'_j(s_i)=[L_{\max j}-L_j(s_i)]^{-\eta},\eta\in(0.5,1.5) \tag{3}$$

由正向负载指标可得权重参数 v_j 可以用式(4)计算：

$$v_j=\frac{L'_j(s_i)}{\sum_{j=1}^{4}L'_j(s_i)} \tag{4}$$

由式(3)—式(4)可知，当负载指标 $L_j(s_i)$ 过高时，其正向负载指标 $L'_j(s_i)$ 以及权重参数 v_j 将被放大，综合负载率 $L(s_i)$ 随之增大，LB 将减少任务分配，从而避免集群内的节点因某一项指标过载导致节点故障。正向负载指标变化速率取决于 η 的取值，η 将影响布谷鸟种群初始化。

本文负载均衡算法的调度目标是将 LB 上的所有任务根据负载分配权重调度至集群节点上，以平衡节点间的负载。节点 s_i 负载分配权重可以用式(5)计算：

$$W(s_i)=\frac{P(S_i)[1-L(s_i)]}{\sum_{j=1}^{m}P(s_j)[1-L(s_j)]} \tag{5}$$

其中：$P(s_i)[1-L(s_i)]$ 为在考虑综合负载后节点当前剩余的服务容量；分母为集群当前的总服务容量。

本文在进行负载分配时，充分考虑 CPU、内存、磁盘 I/O、网络实时状态对节点任务处理能力的影响。当任务序列不为空时，基于当前负载状态计算负载分配权重，并以此指导本次任务调度；当集群负载变化时，重新计算新的负载分配权重，用于指导下一次任务分配，从而形成了基于自适应分配权重的请求任务调度。负载分配权重模型最终将产生一组任务分配方案作为布谷鸟搜索算法的初始种群。

1.3　任务调度时间模型

负载分配的目标是均衡各节点间的负载，最小化集群的任务调度时间。

从 LB 任务调度序列的入口任务 t_{start} 开始执行到出口任务 t_{end} 结束的时间段称为节点任务的完成时间。由于请求任务的性质不尽相同，对服务器资源的占用也不相同，请求任务集合在集群中的执行时间矩阵可以用式(6)计算：

$$ET(t_i,s_j)=\begin{bmatrix}ET_{11} & \cdots & ET_{n1}\\ \vdots & \ddots & \vdots\\ ET_{1m} & \cdots & ET_{nm}\end{bmatrix} \tag{6}$$

其中：ET_{ij} 为任务 t_i 在节点 s_j 上的执行时间。

假设 LB 调度至节点 s_j 的任务集合为 $T(s_j)=\{t_{j,1},t_{j,1},\cdots,t_{j,l}\}$，$l$ 表示任务个数。则节点 s_j 上任务最早开始时间 $EST(t_{j,k},s_j)$ 和最早结束时间 $EFT(t_{j,k},s_j)$ 可以用式(7)—(8)计算：

$$EST(t_{j,k},s_j)=\begin{cases}p(t_{j,k}),t_{j,k}=t_{j,\text{start}}\\ E_{\max}=\begin{cases}p(t_{j,k})\\ AST(t_{j,k})\end{cases}\end{cases} \tag{7}$$

$$EFT(t_{j,k},S_j)=\sum_{k=1}^{l}ET_{kj}+EST(t_{j,k},S_j),\quad k=1,\cdots,l \tag{8}$$

其中：$p(t_{j,k})$ 为节点 s_j 与 LB 通信以及加载任务序列的时间；$AST(t_{j,k})$ 为任务 $t_{j,k}$ 在节点 s_j 上的实际开始时间；$E_{\max}$ 为 $p(t_{j,k})$ 与 $AST(t_{j,k})$ 中的最大值。整个任务序列在集群中的完成时间可以用式(9)计算：

$$DT=\max\{EFT(t_{j,k},s_j)\} \tag{9}$$

本文算法进行负载均衡调度时以集群任务序列的最晚完成时间最小为目标,因此其目标函数定义如式(10)所示:

$$fitness=\min(DT) \tag{10}$$

2 改进的布谷鸟搜索算法

本文引入混沌变异和反向学习该进布谷鸟算法,并将改进的算法与负载分配模型结合,搜索最优负载分配权重。

2.1 混沌变异

混沌运动是特定范围内一种非线性现象,可按照一定规律且不重复地遍历所有状态,将混沌因子根据特定的动力学模型映射到寻优的取值区间,因此混沌可以使算法在寻优时避免陷入局部最优。论文算法引入混沌优化策略,并以 Logistic 映射作为混沌信号发生器,Logistic 迭代公式可以用式(11)表示:

$$y_i=\mu y_i(1-y_i),i=1,2,\cdots,k0<\mu\leqslant 4 \tag{11}$$

其中:μ 为控制参数;y_i 取值区间为(0, 1);k 为混沌序列的长度。当 $\mu=4$ 时,呈现典型混沌特征,具有随机性、规律性、遍历性和对初值敏感性等[10]。

2.2 反向学习

反向学习(Opposition-based learning, OBL)是智能计算领域中的一种技术。该算法基于当前的可行解评估其反向学习的解,并选择更好的可行解[14]。

定义1 若 x 是区间$[a, b]$的任意实数,则 x 的基本反向学习的数公式可以用式(12)表示[15]:

$$x'=a+b-x \tag{12}$$

由此可得,x 到 a 的距离为$|x-a|=x-a$,根据式(12)可得 x'到 a 的距离为$|b-x'|=|b-(a+b-x)|=|a-x|=x-a$,因此,这两个距离是相等的。任意实数与他的反向学习数示意图如图2所示。

图2 任意实数与他的反向学习数示意图

定义2 若 $p=(x_1, x_2,\cdots, x_D)$之间的任意 D 维空间的点 x_i 的取值区间为$[x_{\min}, x_{\max}]$,则它的反向学习的解可以用式(13)计算[16]:

$$\begin{cases}p'=(x'_1,x'_2,\cdots,x'_D)\\ x'_i=x_{i,\min}+x_{i,\max}-x_i\end{cases} \tag{13}$$

可以看出,由于可行解和反向学习的点位于搜索空间的两侧,类似于同时两个对称空间,可扩大解的搜索空间、提高搜索效率。

2.3 改进布谷鸟搜索算法

CS算法是一种启发式算法[7-9]。CS算法初始化参数相对较少,且迭代寻优效率较高。根据CS算法的理想状态[8],算法初始化后通过 Lévy 飞行更新当前巢穴的位置,用公式可以表示为:

$$x_i^{t+1}=x_i^t+\alpha Levy(\beta) \tag{14}$$

其中:x_i^{t+1} 和 x_i^t 分别为第 i 个鸟窝位置在第 $t+1$、t 代中的位置;n 为鸟窝数量;α 为步长缩放因子,通常为 0.01[17]。

巢穴更新后,用区间(0, 1)均匀分布的随机数 R 与蛋被发现概率 P_a 作比较,若 $R>P_a$,则丢弃不好的巢穴位置,并根据式(15)构造新的巢穴位置[7];反之,保留当前位置。P_a 的取值区间是(0, 1),表示宿主发现宿主巢中外来鸟蛋的概率,大部分研究者将 P_a 设为 0.25[18]。鸟窝位置的更新公式可以表示为:

$$x_i^{t+1}=x_i^t+r(x_j^t-x_k^t) \tag{15}$$

其中:r 为比例因子,是区间(0, 1)上的均匀分布随机数;x_j^t 和 x_k^t 为 t 代中的两个随机解。

布谷鸟搜索算法一般使用随机函数生成初始化种群,随机函数易导致初始鸟巢的离散程度低,在解空间中的位置不够均匀。这使得布谷鸟种群迭代寻优时易陷入局部最优且收敛缓慢,从而影响集群在高负载情况下的负载均衡效率。

由1.1知,任务集合 T 被调度至节点集合 S,用 $P=\{p_1, p_2,\cdots, p_i,\cdots, p_n\}$表示一种调度方案,其中 p_i 表示第 i 个任务调度至序号为 p_i 的节点。用随机函数生成规模为 N 的初始种群 $X_0=(x_1, x_2,\cdots, x_i,\cdots, x_N)$,其中 x_i 表示一种调度方案 P。

本文引入 Logistic 混沌信号对初始种群解进行初始化,并在算法迭代过程中每一轮产生的劣解进行混沌变异更新,增加种群迭代过程中解的多样性。首先,使用适应度函数对种群中的布谷鸟进行评估,从适应度值排名前 6%的群体中随机选择一个解 l'_{best},与最优解 l_{best} 进行混沌变异产生新解 y_i,变异公式如式(11)所示。然后,将新解映射为混沌变量 Y_i,再将 Y_i 与 l_{best} 线性叠加,生成混沌变异变量 X_{best},映射公式可以用式(16)表示:

$$\begin{cases}Y_i=LB+y_i(HB-LB)\\ x=(1-\delta)l_{best}+\delta Y_i\end{cases} \tag{16}$$

其中:$i=1,2,\cdots,k$;HB、LB 表示混沌向量定义域的上、下限;δ 为调整系数,根据算法迭代信息确定,其具体公式可以用式(17)表示:

$$\delta=\frac{T_{\max}-t+1}{T_{\max}} \tag{17}$$

其中：t 为当前迭代次数；$T_{\max}$ 为最大迭代次数。

为减少 CS 算法迭代过程中各维度间的干扰，本文不使用整体更新再评价的方式，通过引入反向学习策略，对更新后的解的每一维进行反向学习。将更新的各维度值映射到它的反向学习数进行空间搜索，淘汰退化维度信息，增加解空间的搜索范围，从而提高了搜索效率和寻优能力。

设种群规模为 N，维度为 D，迭代过程选择更新位置时的位置矩阵为 $X_{N\times D}$，每个维度的上、下限分别为 $HB_{N\times D}$、$LB_{N\times D}$。为了更加充分地遍历搜索空间，在动态的搜索空间计算反向值，计算反向值公式可以用式(18)表示：

$$X^{*}_{N\times D}=k(HB_{N\times D}+LB_{N\times D})-X_{N\times D} \tag{18}$$

在式(18)中引入式(1)，得到式(19)：

$$X_i^{t+1}=t(X_{\text{best}}+X_{\text{worst}})-X_i^t \tag{19}$$

其中：t 的取值空间为 $[step_{\min},\ step_{\max}]$，$step_{\min}$、$step_{\max}$ 分别为 Lévy 飞行的缩放因子[19]。

以下给出基于混沌变异和反向学习改进的布谷鸟搜索算法：

步骤 1：设置鸟群规模 N，最大迭代次数为 $T_{\max}$，发现概率 P_a，结束条件，设定最小步长 $step_{\min}$ 和最大步长 $step_{\max}$。

步骤 2：根据混沌变异初始化规模为 N 的鸟巢位置 $X_i=(x_1,\ x_2,\cdots,\ x_N)$。

步骤 3：在当前种群 X_i 中，计算种群中每个解的适应度函数值，并寻找当前的最优调度策略；

步骤 4：根据精英保留策略保留当前最优解，同时按式(14)对种群内的鸟巢位置进行 Lévy 飞行更新，得到个体的新位置 x_i^{t+1}，计算适应度并替换。

步骤 5：用(0，1)上均匀分布的随机数 r_i 与鸟蛋被发现概率 P_a 作比较，若 $r_i>P_a$ 则丢弃当前解，根据式(16)进行混沌变异产生新解；反之，保留当前解。计算解的适应度，排序找到适应度较好的鸟巢。

步骤 6：对步骤 5 中保留下来的解，根据式(19)学习，分别考虑各维度的更新信息。组合当前维度反向学习后与其他维度的值得到新解，评价新的组合解。若改善当前解的质量，则保留当前维度的更新结果；反之，则放弃当前维度的更新。仍然利用精英保留策略进行下一维度的反向学习更新，直到各维度更新完毕。

步骤 7：满足最大迭代次数或达到停止条件时，跳转到步骤 8，否则跳转到步骤 4 进行下一轮搜索。

步骤 8：根据最优解输出最优负载分配权重。

2.4　算法流程

论文算法的总体流程如图 3 所示，LB 负责收集节点服务器的负载信息，结合负载分配权重模型输出分配权重集，并根据集群实时状态对权重进行自适应调整，经混沌变异后作为布谷鸟搜索的初始种群，使用反向学习策略加速最优分配权重的输出，LB 根据最优权重进行任务调度。

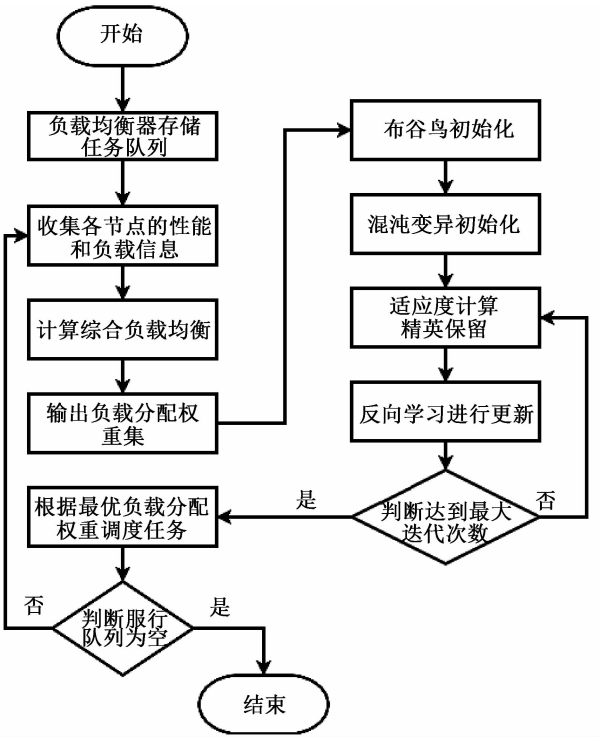

图 3　基于改进布谷鸟搜索的自适应负载均衡算法流程

布谷鸟搜索算法的时间复杂度由目标函数的时间复杂度 $fitness(n)$ 计算决定，当 $fitness(n)$ 运算阶数高于 n，时间复杂度为 $O(fitness(n))$；当相等或低于时，时间复杂度为 $O(n)$[20]。根据本文负载均衡算法的流程图 3 可知，改进的布谷鸟算法增加了 $O(nfitness(n))$ 的运算量，为了扩大搜索空间，引入的反向学习增加了一次个体更新，总的时间复杂度公式可以用式(20)表示：

$$T(n)=O(n+nfitness(n)) \tag{20}$$

算法的空间复杂度取决于种群规模 N 和搜索维度 D 的影响，计算公式可以用式(21)表示：

$$S(n)=O(D\times N) \tag{21}$$

本文所提负载均衡算法的算法复杂度随着维度的升高，其时间复杂度和空间复杂度都升高。

3 实验结果与分析

为检验本文所提算法在Web集群负载均衡问题中的效果,论文从两个方面来进行验证,首先是通过仿真实验与基准函数进行比较来验证改进后的算法的性能,其次是通过在集群系统任务调度中的负载均衡效果来进一步验证。

3.1 算法性能测试

仿真测试环境如下:Windows10操作系统,Intel(R) Core(TM) i7-5500CPU 2.40 GHa,8 GB内存,Matlab 2014仿真软件。在仿真实验中使用高维优化算法测试函数对各待测算法进行评估,论文使用的测试函数 f_1—f_4 如表1所示。

表1 4个常见测试函数

函数名	表达式
Rastrigin	$f_1(x)=\sum_{i=1}^{n}[x_i^2-10\cos(2x_i)+10]$
Griewank	$f_2=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$
Ackley	$f_3=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2x_i)\right)+20+e$
Sphere	$f_4=\sum_{i=1}^{n}x_i^2$

其中,4个测试函数设定初始范围如下:

a) Rastrigin取值区间为[-5.12,5.12],此多峰值函数的理论最优解是0,寻优难度较大。

b) Griewank取值区间为[-20.20],此函数理论最优解是0,搜索区域比较大且包含多个极小值点。

c) Ackley取值区间为[-20,20],其理论最优解是0,较多的局部最小值使其寻优难度较大。

d) Sphere取值区间为[-100,100],理论最优值为0,该函数是单峰值函数。

为观察并解析论文算法求解的质量,对论文设定种群规模30,最大迭代次数1000次,被发现概率为0.25,$step_{min}$和$step_{max}$步长因子分别为0.1、1.3,维数 D 取20。基于集群调度对负载均衡算法的要求,论文分别从平均适应度和方差两个方面对CS和ICS-ALB进行比较,不同维度下的实验数据见表2。

根据表2的相关寻优数据可以看出,当维数 $D=20$ 时,ICS-ALB算法在迭代次数和收敛精度上较CS算法有着很显著的提升效果。从解的质量上看ICS-ALB算法整体解的质量高于CS算法,且迭代次数 $TICS\text{-}ALB<T_{CS}$,可知ICS-ALB在后期可以避免在最优解附近震荡的现象,加速最优负载分配权重的输出,从而减少集群整体的服务响应时间。

表2 维度 $D=20$ 时测试函数基于CS和ICS-ALB算法的测试数据

函数	维度	算法	Mean	Variance	迭代次数
Rastrigin	20	CS	4.88	2.51	778
	20	ICS-ALB	2.62	8.14×10^{-1}	313
Griewank	20	CS	5.14×10^{-2}	2.26×10^{-2}	679
	20	ICS-ALB	2.97×10^{-3}	5.73×10^{-3}	261
Ackley	20	CS	1.81×10^{-6}	3.04×10^{-6}	721
	20	ICS-ALB	3.61×10^{-8}	6.27×10^{-8}	301
Sphere	20	CS	2.18×10^{-16}	4.23×10^{-16}	668
	20	ICS-ALB	7.93×10^{-18}	1.74×10^{-18}	236

3.2 负载均衡实验

基于实验室已有的硬件设施,评测环境由1台4核8 GB内存的施压机(Consumer)、1台4核8 GB的LB和5台节点服务器(S_{r1}, S_{r2}, S_{r3}, S_{r4}, S_{r5})组成,节点服务器的配置见表3。

表3 节点服务器配置

节点	CPU/GHz	内存/GB	磁盘速率/(Mb·s^{-1})	带宽/(Mb·s^{-1})	类型
S_{r1}, S_{r2}	2.4	4	50	50	RS
S_{r3}	2.4	8	50	100	RS
S_{r4}, S_{r5}	1.8	4	50	4	LXC

S_{r1}、S_{r2}、S_{r3}为局域网真实服务器(Real server, RS);S_{r4}、S_{r5}是在云主机上构建的Linux容器(Linux container, LXC),其在生产环境中应用较为广泛。对基于WLC、CS和ICS-ALB算法进行集群任务调度负载均衡实验和分析,设定集群节点服务器各指标过载临界值为 $L_{max}=\{0.90, 0.85, 0.85, 0.90\}$。实验主要步骤如下:

a) 集群系统启动后,Consumer向LB发送客户端请求,先进行20 s预热使集群达到进入稳定的负载状态。之后每隔20 s请求发送频率达到一次峰值,峰值持续20 s然后回落至稳定状态,且随着时间推移峰值增大,一次测试持续200 s,以此来模拟高并发不稳定网络环境。施压机请求发送频率变化情况如图4所示。

b) 在LB上通过排队线程接收客户端请求,高并发进入集群的请求被添加到任务队列,LB分别使用WLC、CS和ICS-ALB三种算法进行任务调度。分布式集群架构的核心在于对高并发网络请求的处

理能力，因此集群服务容量的大小是衡量负载均衡算法性能的重要指标。实验以各算法运行过程中集群每秒处理的事务数（Transactions per second，*TPS*）采样来量化集群服务容量，每 20 s 采样一次，如图 5 所示。从图 5 可以看出，随着请求任务的增加，ICS-ALB、CS、WLC 算法的集群 *TPS* 整体呈现上升趋势，在不同程度上实现了负载均衡。当施压频率达到峰值时，各算法的集群 *TPS* 大小基本呈现 $TPS_{ICS\text{-}ALB} > TPS_{CS} > TPS_{WLC}$ 的关系，且 ICS-ALB 算法攀升较快。当施压频率回落至稳定状态，各算法对应的集群 TPS 基本呈现 $TPS_{ICS\text{-}ALB} < TPS_{CS} < TPS_{WLC}$，且 ICS-ALB 算法回落较快，迅速向稳定的负载状态靠近。因此，在面临高并发不稳定的网络状况时，ICS-ALB 可以更好地提升集群服务容量，保证集群的服务质量。

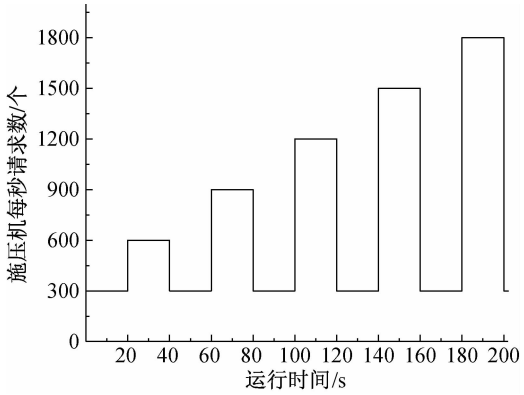

图 4　施压机请求发送频率变化情况

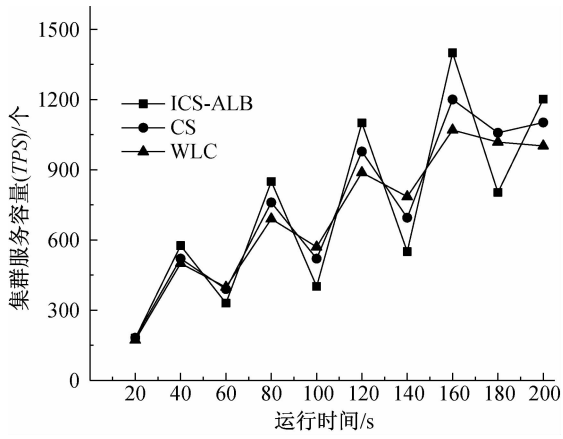

图 5　不同算法的集群服务容量 *TPS* 曲线

Consumer 定时采样 ICS-ALB、CS、WLC 算法的集群平均响应时间如图 6 所示。WLC 算法受高并发网络请求的影响较为明显，当施压频率处于峰值时，WLC 算法的集群响应时间明显增加。ICS-ALB 算法受施压频率影响较小，其平均响应时间整体波动较小且优于 CS 算法，高并发情况下始终保持 2 s 以内的平均响应时间，比 WLC 算法减少 60%左右，其阻塞率明显小于其他两种算法。通过实验发现，引入反向学习和混沌变异算子的布谷鸟搜索算法在集群负载均衡上的应用可有效提高集群高并发服务容量，保持较短的请求响应时间。

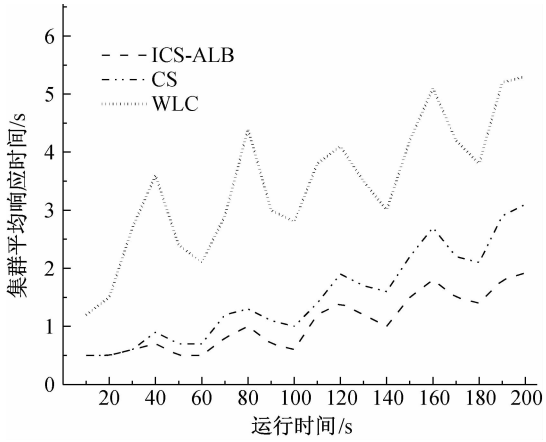

图 6　不同算法的集群平均响应时间曲线

4　结　论

为了解决 Web 集群负载分配时存在节点负载不均衡的问题，综合考虑集群节点的性能和负载，建立了自适应负载分配模型。针对任务分配权重的寻优效率问题，引入混沌变异和反向学习策略，提出了基于改进布谷鸟搜索的优化算法。通过实验，与 CS、WLC 算法相比较，ICS-ALB 算法使集群具有更高的吞吐量和更短的服务响应时间。考虑到更复杂的异构集群环境，论文将在现有研究的基础上，进一步结合节点资源的负载均衡进行深入研究。

参考文献：

[1] 王嘉豪，蔡鹏，钱卫宁，等. 集群数据库系统的日志复制和故障恢复[J]. 软件学报，2017，28(3)：476-489.

[2] 蔡嵩，张建明，陈继明，等. 云计算环境中基于朴素贝叶斯算法的负载均衡技术[J]. 计算机应用，2014，34(2)：360-364.

[3] 王文博，叶庆卫，周宇，等. 基于排队论综合指标评估的动态负载均衡算法[J]. 电信科学，2018，34(7)：86-91.

[4] Li S，Wang F，Xiao B J，et al. Study of load balancing technology for EAST data management [J]. Fusion Engineering and Design，2014，89(5)：750-753.

[5] Sun D H，Zhu Q. Comparison and improvement of load balance scheduling algorithm based on cluster technology [J]. WIT Transactions on Engineering Sciences，2014，92：67-74.

[6] 郑浩，李宁，杨小涛. 自适应负载指标权值的负载均衡算法[J]. 计算机工程与设计，2019，40(3)：623-626.

[7] Yang X S，Deb S. Cuckoo search via lévy flights[C]// 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC). Coimbatore，India：IEEE，2009：210-214.

[8] Yang X S，Deb S. Engineering optimisation by cuckoo search [J]. International Journal of Mathematical Modelling and Numerical Optimisation，2010，1(4)：330-343.

[9] Yang X S，Deb S. Multiobjective cuckoo search for design optimization [J]. Computers & Operations Research，2013，40(6)：1616-1624.

[10] 战非，张少茹. 适应云计算的混沌布谷鸟算法应用优化研究[J]. 控制工程，2017，24(7)：1486-1492.

[11] 李佳，夏云霓. 云计算资源调度问题求解的布谷鸟搜索算法[J]. 控制工程，2019，26(1)：170-174.

[12] 张光亚. 基于ICS的云计算资源预测负载的研究[J]. 科技通报，2019，35(4)：93-98.

[13] 赵博颖，肖鹏，张力. 基于混合并行布谷鸟搜索的作业调度算法[J]. 计算机工程与设计，2019，40(3)：719-724.

[14] Wei W H，Zhou J L，Chen F，et al. Constrained differential evolution using generalized opposition-based learning[J]. Soft Computing，2016，20(11)：4413-4437.

[15] Zhang S，Luo Q F，Zhou Y Q. Hybrid grey wolf optimizer using elite opposition-based learning strategy and simplex method [J]. International Journal of Computational Intelligence and Applications，2017，16(2)：1750012.

[16] Ahandani M A，Alavi-Rad H. Opposition-based learning in shuffled frog leaping：an application for parameter identification [J]. Information Sciences，2015，291：19-42.

[17] Jiang M L，Luo J Y，Jiang D D，et al. A cuckoo search-support vector machine model for predicting dynamic measurement errors of sensors [J]. IEEE Access，2016，4：5030-5037.

[18] Naik M，Nath M R，Wunnava A，et al. A new adaptive Cuckoo search algorithm[C]//2015 IEEE 2nd International Conference on Recent Trends in Information Systems (ReTIS). Kolkata，India：IEEE，2015：1-5.

[19] Cai Z F，Yang X D. Cuckoo search algorithm with deep search[C]//2017 3rd IEEE International Conference on Computer and Communications (ICCC). Chengdu：IEEE，2017：2241-2246.

[20] 肖海林，张文娟，聂在平，等. 基于布谷鸟搜索算法的用户选择和干扰对齐[J]. 电子科技大学学报，2017，46(6)：801-805.

(责任编辑：康 锋)