面向服务可靠性的云资源调度方法

周 平[1,2],殷 波[3],邱雪松[1],郭少勇[1],孟洛明[1]
(1. 北京邮电大学网络与交换技术国家重点实验室,北京 100876;
2. 中国电子技术标准化研究院 云计算标准与应用工业和信息化部重点实验室,北京 100007;
3. 清华大学信息技术研究院 网络大数据技术研究中心,北京 100084)

摘 要: 随着云计算成为重要的信息基础设施,越来越多的应用迁移到云上,云服务的可靠性日益重要,尤其是边缘计算新模式的引入,对云服务可靠性提出了更高的要求.如何通过资源调度保障服务可靠性成为了当前研究的热点.为此,针对云-边协同的应用场景,开展面向服务可靠性的云资源调度方法研究,提出基于马尔科夫预测模型的云资源调度算法,实现节点负载判断、待迁移任务和节点选择、迁移路由的决策,以解决云服务节点失效情况下的任务调度和负载均衡问题,实现快速的云服务故障恢复,提高云服务的可靠性.实验结果表明,本文所提方法能够有效保证节点失效情况下的服务可靠性.

关键词: 云服务;可靠性;资源调度;马尔科夫过程
中图分类号: TP393.1 **文献标识码:** A **文章编号:** 0372-2112 (2019)05-1036-08
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2019.05.009

Service Reliability Oriented Cloud Resource Scheduling Method

ZHOU Ping[1,2], YIN Bo[3], QIU Xue-song[1], GUO Shao-yong[1], MENG Luo-ming[1]
(1. *State Key Laboratory of Network and Switching Technology, BUPT, Beijing* 100876, *China*;
2. *Key Laboratory of Cloud Computing Standards and Applications, Ministry of Industry and Information Technology, China Electronics Standardization Institute, Beijing* 100007, *China*;
3. *Research Center of Network Big Data Technology, Institute of Information Technology, Tsinghua University, Beijing* 100084, *China*)

Abstract: As Cloud Computing becomes an important information infrastructure, more and more applications are being migrated to the cloud. Therefore, the reliability of cloud services becomes increasingly important. In particular, the introduction of new edge computing mode puts forward higher requirements on the reliability of cloud services. How to guarantee the reliability of services through resource scheduling has become a hot topic of current research. In Cloud-Edge collaborative application scenarios, we research on a service reliability oriented cloud resource scheduling method to support cloud service reliability. And the cloud resource scheduling algorithm based on markov prediction model is put forward to solve the problem of task scheduling and load balancing in cloud service node failure situation, including the judgment of node load degree, the selection of migrated task and nodes, and the decision of migration routing. The goal is to achieve rapid cloud service recovery and to improve the reliability of cloud services. The experimental results show that the proposed method can effectively guarantee the service reliability.

Key words: cloud service; reliability; resource scheduling; Markov process

1 引言

在云计算提供服务环境[1]中,通过与边缘计算结合,能够更好地针对当前万物互联环境下,业务数据具有高实时性、数据量大的特点,提高业务处理能力,加快响应速度.上述环境对云服务可靠性提出了更高的要求[2],边缘计算节点如何解决突发故障以有效满足全局业务的处理需求,是当前云服务可靠性[3]面临的主要问题之一.

边缘计算改变了传统云计算将数据上传至计算中心统一调度和分析的做法,终端节点首先通过边缘层的数据采集与监控系统,直观了解设备的运行状态.当

收稿日期:2018-10-17;修回日期:2019-03-01;责任编辑:郭游
基金项目:国家自然科学基金(No.61702048);国家重点研发计划(No.2018YFB1004200)

设备出现故障、指标出现异常时,终端节点根据事先设定的各项参数,自行解决部分计算任务,将问题在本地进行处理,有效降低数据上传到云端处理的时延,大幅提高服务质量和用户体验. 然而,当设备发生故障时,数据会发生突增,本地终端可能无法及时处理数据从而导致数据丢失,引发严重的可靠性问题. 为了及时处理故障以及上传数据,本地终端将向上层的边缘计算服务器发送请求帮助的消息,避免大业务量导致的高延迟响应和超出本地缓冲区的业务丢失. 因此,设计资源调度算法,实现在边缘设备节点或云端节点出现故障时,有效保证云服务的可靠性,是亟待解决的问题.

针对云计算中的资源调度问题,文献[4]提出了基于免疫克隆的偏好多维 QoS 启发式算法,该算法对用户偏好和 QoS 要求进行量化,优化 QoS 目标函数从而提高了最优解的收敛能力. 在分布式资源调度领域,文献[5]提出了一种 Hadoop Yarn 的资源调度方法,该方法融合蚁群算法和粒子群算法,优化蚁群算法的信息素挥发系数并融合粒子群算法的自我认知能力,最终提高解的精度. 文献[6]在 P2P 网络上提出了一种资源协作共享策略,文中根据排队理论为资源的提供者,构建层次型资源调度模型,实验结果表明该方法提高了资源服务能力,降低了资源请求丢失率. 文献[7]中提出了带有业务分区的混合资源调度算法. 针对异构资源调度问题,文献[8]提出了一种任务共享的解决方案,一定程度上减小了调度的费用开销. 文献[9]针对资源调度模型通用性不强的问题,提出了基于边际效用函数的优化解决方案. 然而,上述方法分别从不同视角提出自己对云计算资源调度问题的解决方案,但都没考虑到云服务节点失效的情况,本文在研究分析现有的资源调度问题解决方案上,提出了基于马尔科夫预测模型的调度算法,解决云服务节点失效情况下任务调度和负载均衡问题,并提供快速的云服务故障恢复功能,从而提高云服务质量的可靠性.

主要组织结构如下:在第二部分对问题进行了建模,第三部分提出了面向服务可靠性的云资源调度方法,第四部分对所提方法进行了实验验证,第五部分进行了全文小结.

2　问题描述

本文所解决方法的典型业务场景是负载均衡网络,该场景的结构拓扑图如图 1 所示,分为终端侧、边缘侧和云侧三个部分. 该拓扑图的核心在于终端侧和边缘侧[10]. 终端节点是一系列传感器,能够进行采样监控数据的初步筛选与处理,数据量大时上传至边缘侧. 边缘侧包括若干服务器和基站,以及左右两个服务器.

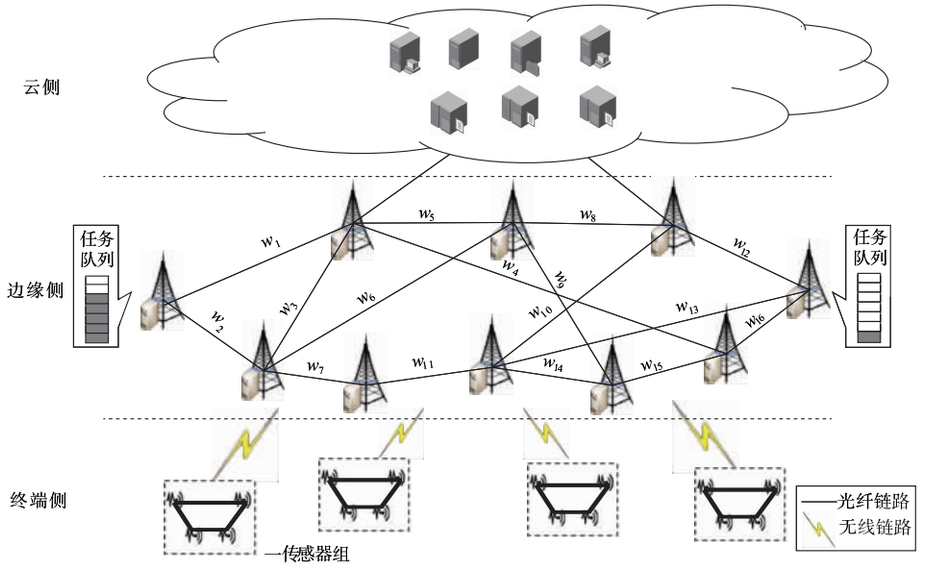

图1　网络拓扑结构图

图 1 中展示了边缘侧设备各自的任务队列,其中灰色条代表着当前服务器已经处于等待状态的任务数,白色条代表着服务器当前状态下的剩余容量. 整个系统业务流程为终端侧收到预警信息后请求边缘服务器协助计算,边缘服务器处理完毕后将数据上传至云服务器[11]. 在整个过程中,每个边缘计算服务器拥有一个任务等待队列,当任务等待队列达到高负载状态后,将由于故障突发的业务负载至低负载服务器[12]. 边缘计算服务器采用分布式组网,通过网络优化算法,最终实现降低任务等待时延,提高边缘计算服务器的使用率,实现云服务的可靠性目标.

3 基于马尔科夫预测的云资源调度算法

在边缘计算环境下,故障突发的业务调度策略流程如图2所示. 终端节点发出数据业务的计算请求后,由于自身资源不足,将从边缘计算服务器获得计算、存储等服务. 本地边缘计算服务器根据自身负载程度判断是否接受该请求. 若超出阈值,则向边缘网络发出负载协作请求,直到发现能够承载业务的低负载服务器接受请求并选择合适的负载转移链路. 经过上述处理过程后,实现了全局流量在业务突发情况下的均衡,保证了每个服务器都在合适的负载范围内[13],避免单台服务器发生超载现象导致服务器性能下降,无法对终端业务请求做出响应. 以此,在实现故障容错的基础上,当前云服务不会因为负载不均导致服务质量的下降,有效提高了云服务的可靠性[14,15].

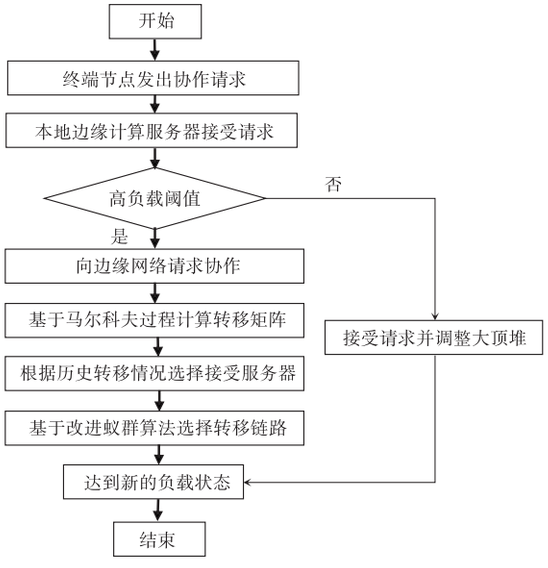

图2 算法流程图

3.1 节点负载程度的判断

假设 $S=\{1,2,3,\cdots,N\}$ 表示所有服务器集合,每个服务器高负载阈值记作 $\mathrm{MAX}(i)$,$i\in S$,低负载阈值记作 $\mathrm{MIN}(i)$,$i\in S$,标准负载阈值记作 $\mathrm{Normal}(i)$,$i\in S$. 单个业务对于服务器的占用指标主要包括CPU使用率 $C(t)$,内存占用率 $M(t)$,任务等待时间 $W(t)$,使用一个系数来表示各参数的重要程度,则 t_1 时刻的任务 i 的负载可以表示为:

$$L_i(t_1)=\lambda_1 C_i(t_1)+\lambda_2 M_i(t_1)+\lambda_3 W_i(t_1) \quad (1)$$

其中,$\sum\lambda=1$. 整个服务器的负载记为 $\mathrm{LOAD}(i)$,则当前时刻负载程度可以表示为:

$$\mathrm{LOAD}(i)=\sum_{t=1}^{n} L_i(t) \quad (2)$$

服务器将根据当前时刻负载分为以下几种情况:

(1)当 $\mathrm{MIN}(i)\leq\mathrm{LOAD}(i)\leq\mathrm{MAX}(i)$ 时,此时服务器处于正常状态,服务器能够正常处理终端侧发送的任务请求并分配相应的资源进行计算存储,系统能够正常响应用户请求,无滞后现象,并未出现超载情况.

(2)当 $\mathrm{MAX}(i)<\mathrm{LOAD}(i)$ 时,服务器负载过高,系统资源满载,用户响应严重滞后. 标记该节点为高负载节点,开始进行下一阶段的负载转移过程. 由局部性原理可知,该服务器大概率将在之后的一段时间内处于高负载状态,因此需要转发业务至其他低负载服务器直到 $\mathrm{LOAD}(i)\leqslant\mathrm{NORMAL}(i)$,从而使服务器下一时段处于正常状态.

(3)当 $\mathrm{LOAD}(i)\leqslant\mathrm{MIN}(i)$ 时,服务器处于低负载状态,系统资源空闲,状态稳定. 标记该节点为低负载节点,等待接下来可能出现的协作请求,服务器任务等待队列将用于接受高负载节点的任务.

3.2 待迁移任务的确定

由上文可知,任务的负载状况分为CPU使用率 $C(t)$,内存占用率 $M(t)$,任务等待时间 $W(t)$ 三个部分. 对于本地服务器来说,一般认为高负载节点往往希望迁移负载较大的任务,避免负载较大的任务在等待队列中等待时延过长. 如图3所示,本文将对服务器采用堆结构来管理收到的任务,即所有任务按照任务大小构成大顶堆,堆顶元素为所有任务中最大的负载量. 当堆顶元素被迁出或有新的任务进入到本地服务器时,节点将自动调节堆结构,保证堆顶任务始终是剩余任务中负载量最大的. 这样每次发生负载迁移时都迁出堆顶的任务,以降低本地服务器的负载程度.

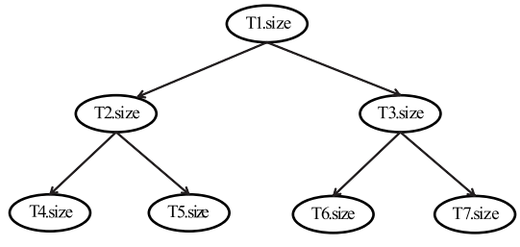

图3 任务大顶堆结构图

任务队列建立堆结构示意图如图3所示,Ti. size表示任务. 从叶子节点开始,自底向上,形成大顶堆,并且对堆结构进行调整,将每个非叶子结点都进行调整,调整顺序为从底层至顶层,调整过程中含有递归,由此二叉树成为一个大顶堆或小根堆.

3.3 待迁移节点的确定

现有云计算系统中,通常采用集中式资源调度策略. 系统中设置收集各节点负载信息的节点,该节点根据其他节点发送来的调度请求进行故障资源恢复,再将执行后的分配信息发送给各个节点,进行负载均衡

优化. 本文所采用的分布式资源管理系统,将在每一个边缘服务器上应用统一的资源优化策略. 当高负载节点接收到故障请求需要对当前业务负载进行转移时,对于已经确定的待迁移任务,根据节点历史分配的序列,得到其余低负载节点被分配到任务的概率,并把当前需要迁移的任务分配到概率最高的节点,同时被分配节点在接受待迁移任务后不会发生高负载现象. 由于边缘侧各服务器之间相互无关联,历史节点序列的两个相邻节点也具有无关联性. 因此,本文将采用马尔科夫预测模型来确定待迁移节点.

马尔科夫过程是指随机过程$\{X_n, n=1,2,\cdots\}$的构造使得X_{n+1}的条件概率分布仅仅依赖X_n的值而与之前的值无关. 状态空间$X=\{X_i, i=1,2,\cdots\}$用概率表示为:

$$P\{X_{n+1}\mid X_1=x_1, X_2=x_2\cdots\}= P\{X_{n+1}=x_{n+1}\mid X_n=x_n\} \tag{3}$$

转移概率表示为:

$$p_{ij}(m)=P\{X_{n+1}=x_{n+1}\mid X_n=x_n\} \tag{4}$$

其中m为非负整数,称$p_{ij}(m)$为$\{X_n\}$在时刻m从状态i出发到达j的转移概率. 状态向量分布$\pi=\{\pi_1,\pi_2,\pi_3,\cdots\}$表示当前时刻各个状态出现的概率,在$n$时刻状态$i$转移到$n+1$时刻状态$j$的概率$p_{ij}$组成概率转移矩阵$A$. 一个包含$n$个状态的可遍历的马尔科夫链,在$n$时刻的状态分布向量公式为:

$$\pi(n)=\pi_{(n+1)}A \tag{5}$$

$Max(\pi_i)$对应的i值即为预测n时刻任务转移最可能的状态.

将本地服务器占有率用百分比量化后,可知每个服务器的负载程度均在0% ~100%间. 由上文所计算得到的服务器负载程度,可将服务器的负载程度划分为0~9共十个状态等级,具体划分函数为:

$$State(i)=\lfloor Load(i)/10\rfloor \tag{6}$$

由式(6)即可得到当前时刻下,边缘网络所有服务器的马尔科夫过程的状态空间$E=\{0,1,2\cdots,9\}$. 对高负载服务器来说,进行负载迁移时,是以任务为单位分配到其他各个服务器. 一个任务被分配到状态等级为i的节点,记下该节点的状态号i,这样就能构成一个状态序列,该状态序列即可被看作是一条马尔科夫链. 因此,当该服务器新的高负载情况出现时,利用式(3)即可得到最大概率转移节点,并判断转移后接受节点负载状况是否处于正常状态. 若正常,则进行转移,否则继续搜索新的接受节点.

考虑当前时刻为t,此时节点i出现高负载情况,t时刻节点i的负载状态为X_t,需要迁移若干任务使得服务器正常. 前n个时刻($n<t$),负载转移序列为$X_1,X_2,X_n\cdots$,从这个转移序列中,把相邻时刻状态转移$i\rightarrow j$出现的次数记作C_{ij},可以得到i状态向j状态的转移概率p_{ij},

$$p_{ij}=\begin{cases}\dfrac{C_{ij}}{\sum C}, & C_{ij}\neq 0\\ 0, & C_{ij}=0\end{cases} \tag{7}$$

由状态转移概率组成马氏链的转移概率矩阵为:

$$\boldsymbol{P}=\begin{bmatrix}p_{11} & p_{12} & \cdots & p_{19}\\ p_{21} & p_{22} & \cdots & p_{29}\\ \cdots & \cdots & \cdots & \cdots\\ p_{91} & p_{92} & \cdots & p_{99}\end{bmatrix} \tag{8}$$

负载转移仅仅发生在高负载状态向低负载状态. 因此,该转移概率矩阵是一个下三角矩阵. 把状态j出现的次数记为$N(j)$,则马尔科夫链在$t-1$时刻的状态分布向量为:

$$\pi_j=\begin{cases}\dfrac{N(j)}{n}, & N(j)\neq 0\\ 0, & N(j)=0\end{cases} \tag{9}$$

在系统试运行阶段,记录前n个高负载时刻数据的分配. 从第n个时刻开始记为t_0并把t_0作为马尔科夫链的起始时刻. 取初始状态分布向量$\pi^{(0)}=(1/n,1/n,\cdots,1/n)$. 根据公式即可确定状态转移矩阵.

3.4　迁移路由的选择

当确定了迁移起点和迁移终点后,下一步需要解决的是两点间的最短路径问题. 蚁群算法在求解最短路径问题中表现出较好的性能,但也会出现局部最优,易停滞现象. 本文在蚁群算法的基础上,对经典蚁群算法的局限性加入马尔科夫过程,根据历史链路选择数据对初始信息素的不同浓度进行设置,并且每个时刻实时更新全局信息素浓度,大大提高最短路径搜索效率.

边缘网络中节点间的连通往往会有多条路径. 对于两个直连节点来说,如果节点间能够通过其他节点达到的可连通路径越多,则表明可以选择替代直连路径的其他非直连的路径概率就越大,则直连路径的重要性就越小,即链路权值越小. 因此,边缘网络链路的权值与可达到性成反比关系. 如图4所示,i节点到j节点的路径有n条,其中只有$p1$是直连路径.

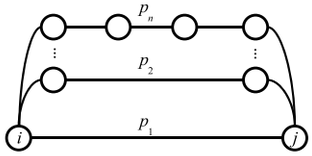

图4　相邻节点的连通性说明图

网络拓扑结构通常用无向图$G=(V,E)$表示,其中V表示网络中节点的集合,包括边缘侧服务器与其基站,$|V|=n$,E表示边缘侧所有的链路集合,$|E|=m$. 链

路邻接矩阵用 $C=[c_{ij}]$,$i,j\in V$ 表示,其中:

$$c_{ij}=\begin{cases}1, & i\text{ 和 }j\text{ 是直连节点}\\0, & i\text{ 和 }j\text{ 不是直连节点}\end{cases} \tag{10}$$

若想得到邻接矩阵的全连通图,根据马尔科夫 C-K 方程可知,n 步转移概率是一步转移概率的 n 次方,如下式所示:

$$P^{(n)}=P^n \tag{11}$$

系统状态的一步转移矩阵由一步转移概率 $p_{ij}^{(1)}$ 组成,对于一步转移概率的分配,我们考虑的参数是相同节点的链路可用带宽比. 设 T_i 为 i 节点的任务出度序列,B_i 为以该节点作为出度下的链路可用带宽,则分配到每一条链路的转移概率为:

$$P_i=\frac{T_i*B_i}{\sum_{i\in m}T_i*B_i} \tag{12}$$

由 C-K 方程可以得到多步转移概率 $P^{(n)}$,由于链路可达性与权值成反比,节点 i,j 的非一步转移概率和为:

$$P_{ij}=\frac{\sum_{t=2}^{n}(P_{ij}^{(t)}+P_{ji}^{(t)})}{2} \tag{13}$$

将链路可达概率归一化,即可得到链路权值为:

$$\varepsilon_{ij}=\frac{\frac{1}{p_{ij}}}{\sum\frac{1}{p_{ij}}} \tag{14}$$

综上所述,整个初始化链路权值分配流程如图 5 所示.

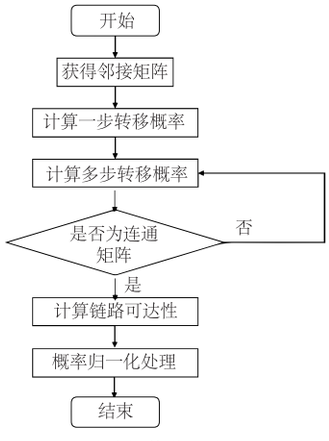

图5 链路权值分配流程图

当获取到链路权值和节点 i 到节点 j 的选择概率后,即为传统蚁群算法里初始化的信息素和蚂蚁转移概率. 将高负载节点的任务分配到低负载节点的过程,即为一个蚂蚁在图上的爬行过程,算法的目标即为找到一条最短路径. 传统蚁群算法往往将初始时刻的各边信息素设为相同值进行策略选择,这种方法容易导致开始时盲目搜索,产生大量无关路径,对局部性更新产生误导. 马尔科夫过程计算出当前时刻下链路权重后,根据链路权重对于各边设置不同的信息素浓度,可以对搜索产生较为合适的方向引导,加快全局最优解的速度,避免陷入局部最优解.

4 仿真与结果分析

通过 MATLAB 进行模拟,对所提基于马尔科夫过程的迁移节点选择和改变初始信息素的改进型蚁群算法进行验证分析,将改进蚁群算法与原始蚁群算法进行了对比.

4.1 实验搭建

为了对基于马尔科夫过程的业务调度策略进行仿真验证,本文共设置了 15 个服务器来组成边缘计算网络,图 6 为此次实验网络拓扑图.

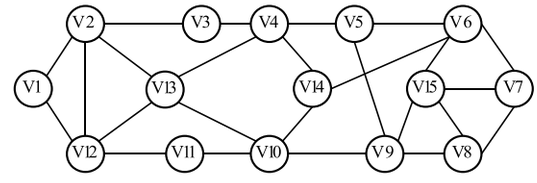

图6 仿真网络拓扑图

其中 V_1 至 V_{15} 表示边缘计算网络中 15 个服务器及其基站,每个服务器和基站都具有一个任务等待队列,服务器间的连线表示相应链路. 此次仿真中改进型蚁群算法的仿真主要实验参数如表 1 所示,设置 50 只随机起点的蚂蚁用来寻找最短距离,除初始信息素浓度外,其他参数与经典蚁群算法一致,利用控制变量法可以更好地对比出该算法的有效性.

表 1 蚁群算法各项参数

参数	数值
m,蚂蚁个数	50
Alpha,信息素重要程度因子	1
Beta,启发函数重要程度因子	5
Rho,信息素挥发因子	0.1
Iter_max,迭代次数	50
Route_best,各代最佳路径长度	-
Length_ave,各代平均路径长度	-

在基于马尔科夫过程的迁移节点选择阶段,将马尔科夫链采用队列的数据结构,对于每一个云服务节点高负载服务器,当需要迁移的任务数据量大时,马尔科夫链利用队列的性质,能够将最先进入队列的序列删除,同时将最新进入队列的任务保留. 在数据流分配阶段,对于 $t-2$ 时刻和 $t-1$ 时刻分别使用两个二维数

组来存储状态转移矩阵.

4.2　实验分析

首先进行迁移节点选择的马尔科夫过程. 在实验开始前,构建一系列马尔科夫序列作为马尔科夫链的初始时刻. 仿真模型根据初始化下的100 次数据分配序列建立了马尔科夫链并计算出状态空间内不同状态间的转移概率. 在进行马尔科夫过程负载迁移后,仿真实验如图 7 所示. 同时从图 7 中可以发现,整个系统的高负载服务器的负载均稳定至正常状态,原有的低负载服务器与高负载服务器的协作有效减少了较大数据流对于均衡程度的破坏. 负载后的结果表明 15 个服务器的负载状态均调整至正常状态以下,说明该算法对于解决负载均衡的问题具有很好的效果.

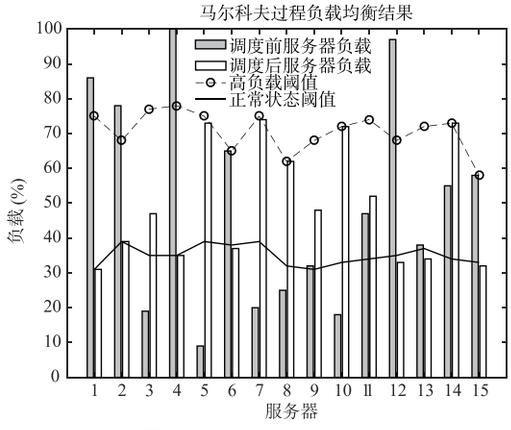

图7　马尔科夫负载迁移结果

通过对单次马尔科夫负载迁移过程进行分析,该过程对于长期低负载节点总有较大的概率去接受高负载节点的任务迁移,进一步增加了低负载节点的接受概率. 对于高负载节点来说,高负载节点能够将任务迁移至低负载节点上,从而降低高负载服务器任务等待时间,达到很好的负载均衡效果. 为了避免单一实验的巧合性和偶然性,本文接下来对该算法进行了 100 次随机重复实验,与单次实验相同,100 次的随机重复实验均建立在一条马尔科夫链上,实验结果如图 8 所示,在 100 次实验中,该算法仅有 10 次无法将高负载服务器均衡至正常状态,负载均衡成功率达到 90%. 异常情况通常是由于该时刻整个系统综合负载较高,绝大部分服务器均处于高负载状态,没有足够数量的低负载服务器来接收高负载的业务. 在这种情况下,边缘侧服务器应告知终端侧发送方减少数据流量的发送. 此外,在大部分情况下,该算法都能将高负载服务器均衡至低负载服务器,实现整个边缘网络负载均衡,说明该负载方案是有效的.

随后进行了从 100 次实验到 500 次实验,这些实验的目的是为了统计基于马尔科夫过程的负载均衡算法

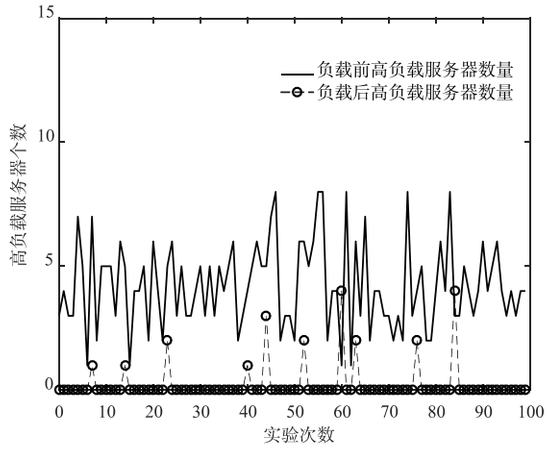

图8　经过100次马尔科夫负载迁移过程结果

的成功率. 经历了大量的实验次数,能够达到负载均衡的所有统计数据如图 9 所示. 由图中可以看出,该算法的成功率始终保持在 80% ~90% 左右. 并且随着实验次数的增加,概率波动性在逐渐减小,最终能够稳定在 85% 左右.

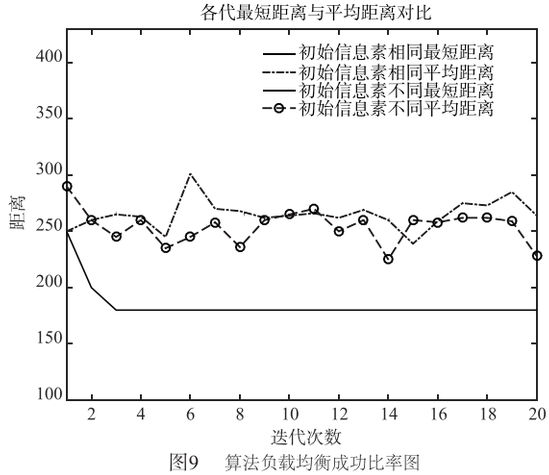

图9　算法负载均衡成功比率图

最后的实验对改进型的蚁群算法进行了仿真,首先根据历次马尔科夫过程的链路选择,计算节点间的一步转移概率,再根据 *C-K* 方程求出多步转移概率并根据选择情况对各链路赋予一定的权重. 最后将链路上计算出的权值作为蚁群算法初始信息素浓度. 对 15 个节点网络进行仿真,仿真实验结果如图 10 所示.

由图 10 可以看出,在经历了 50 次迭代寻找最短路径后,经典蚁群算法在第 35 次迭代就陷入了局部最优解 142. 改进了初始信息素浓度的蚁群算法的最短路径距离稳定在 130,相比于传统的蚁群算法在第 2 次迭代后就更加接近全局最优解. 从平均距离进行分析,50 只随机起点的蚂蚁的平均路径距离也均小于初始信息素浓度相同的情况. 本文所提出的改进型蚁群算法在最

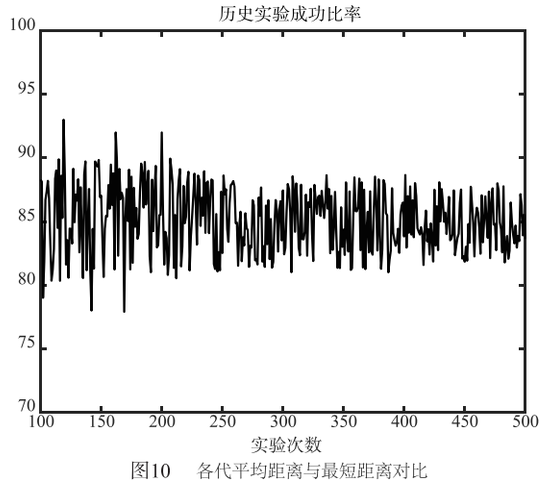

图10 各代平均距离与最短距离对比

优解和平均解上都有明显的改善. 通过改变初始信息素浓度,对历次迭代产生的最优路径进行方向性引导,避免算法陷入局部最优的情况.

5 结论

本文针对边缘计算环境下的云服务可靠性进行研究,提出基于马尔科夫过程的业务调度策略,以解决在边缘节点出现故障时,以及网络流量突发导致的负载不均衡问题. 通过对蚁群算法进行改进,提高了全局路由的优化,缓解了蚁群算法容易陷入局部最优的不足. 最后,通过实验验证了本文所提方法在保障服务可靠性的前提下,对云资源调度的有效性,确保提供更优质的云服务.

参考文献

[1] Mell P, Grance T. The NIST Definition of Cloud Computing[M]. Nova Science Publishers, Inc, 2011. 171 – 173.

[2] Zhou A, Wang S, Cheng B, et al. Cloud service reliability enhancement via virtual machine placement optimization [J]. IEEE Transactions on Services Computing, 2017, 10 (6): 902 – 913.

[3] Zhou A, Sun Q, Li J. Enhancing reliability viacheckpointing in cloud computing systems[J]. China Communications, 2017, 14(7): 1 – 10.

[4] 孙大为,常桂然,等. 一种基于免疫克隆的偏好多维 QoS 云资源调度优化算法[J]. 电子学报,2011,39(8):1824 – 1830.
SUN Da-wei, CHANG Gui-ran, et al. Optimizing multi-dimensional QoS cloud resource scheduling by immune clonal with preference[J]. Acta Electronica Sinica, 2011, 39 (8): 1824 – 2830. (in Chinese)

[5] 李媛祯,杨群,赖尚琦,李博涵. 一种 Hadoop Yarn 的资源调度方法研究[J]. 电子学报,2016,44(5):1017 – 1024.
Li Yuan-zhen, YANG Qun, LAI Shang-qi, LI Bo-han. A study on scheduling method of hadoop yarn[J]. Acta Electronica Sinica, 2016, 44(5): 1017 – 1024. (in Chinese)

[6] 牛新征,周明天,等. 一种应用于移动 P2P 网络的资源协作共享策略[J]. 电子学报,2010,38(1):18 – 24.
NIU Xin-zheng, ZHOU Ming-tian, et al. A cooperative sharing scheme for resources in mobile P2P networks[J]. Acta Electronica Sinica, 2010, 38(1): 18 – 24. (in Chinese)

[7] 熊余,蒋婧,等. TWDM-PON 中带有业务区分的混合资源调度算法[J]. 电子学报,2017,45(6):1490 – 1497.
XIONG Yu, JIANG Jing, et al. Hybrid resource scheduling algorithm with traffic differentiation in TWDM-PON[J]. Acta Electronica Sinica, 2017, 45 (6): 1490 – 1497. (in Chinese)

[8] 田国忠,肖创柏,谢军奇. 一种多 DAG 任务共享异构资源调度的费用优化方法[J]. 电子学报,2014,42(9):1767 – 1774.
TIAN Guo-zhong, XIAO Chuang-bai, XIE Jun-qi. An cost optimization methods for schduling concurrent multiple DAGs sharing heterogeneous resources[J]. Acta Electronica Sinica, 2014, 42(9): 1767 – 1774. (in Chinese)

[9] Stergiou C, Psannis K E, Kim B G, et al. Secure integration of IoT and cloud computing[J]. Future Generation Computer Systems, 2018, 78: 964 – 975.

[10] Aggarwal R. Resource provisioning and resource allocation in cloud computing environment[J]. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 2018, 3 (3): 1040 – 1079.

[11] Madni S, Jamali S. A comparative study of fault tolerance techniques in cloud computing[J]. International Journal of Research in Computer Applications and Robotics, 2018, 6 (3): 7 – 15.

[12] Cimprich P F, DeLuca M G, Cantwell J Q W, et al. Type-to-type analysis for cloud computing technical components with translation scripts: U. S. Patent Application 10/033, 597[P]. 2018 – 7 – 24.

[13] Jhawar R, Piuri V. Fault Tolerance and Resilience in Cloud Computing Environments[M]. Computer and Information Security Handbook (Third Edition), 2017. 165 – 181.

[14] Arpaci I. Antecedents and consequences of cloud computing adoption in education to achieve knowledge management[J]. Computers in Human Behavior, 2017, 70: 382 – 390.

作者简介

周　平　男,1977 年 2 月出生. 2005 年获清华大学软件工程硕士学位,现任中国电子技术标准化研究院云计算标准与应用工业和信息化部重点实验室主任. 主持完成 10 余项包括“核高基”重大专项、重点研发计划、863、工业转型升级在内的重大项目,先后获中国电子学会电子信息科学技术奖二等奖 1 项、北京市科学技术奖三等奖 1 项. 现为北京邮电大学网络与交换技术国家重点实验室博士研究生,主要从事通信与信息系统有关方面的研究.

E-mail:zhouping@ cesi. cn

殷　波　女,1984 年 5 月生,清华大学博士后,北京邮电大学博士. 从事云计算、计算机网络、区块链等研究.

邱雪松　男,1973 年 1 月生,北京邮电大学教授,计算机学院网络管理研究中心主任. 主持完成了 6 项国家和省部级项目,作为项目主研人,参加了 11 项国家级和 5 项省部级项目,取得了一批具有先进水平的成果,其中获国家科技进步二等奖 2 次,省部级科技进步一等奖 3 次,省部级科技进步二等奖 6 次.

郭少勇　男,1985 年 10 月生,2013 年北京邮电大学博士毕业,2013 至 2015 年北京交通大学从事博士后工作. 2015 年任教于北京邮电大学,获得省部级奖项 2 项,主持与参加制定 ITU 国际标准 3 项,发表学术论文若干,主要从事工业与能源互联网领域研究,研究方向为边缘计算、区块链和物联网应用技术.

孟洛明　男,1955 年 5 月生. 北京邮电大学教授、博士生导师,北京邮电大学学术委员会副主任. 曾任北京邮电大学国家重点实验室主任. 长期从事通信网、网络管理和通信软件方面的科学研究和教学工作,主持完成了 10 余项包括国家 863、国家攻关、国家自然科学基金在内的重大项目. 获国家科技进步二等奖 2 次、省部级科技进步一等奖 5 次;研究成果被完整采纳为 7 项国际标准;获国家级有突出贡献的中青年专家、国家杰出青年科学基金获得者、长江学者计划特聘教授等称号.