

doi:10.11835/j.issn.1000-582X.2021.11.006

基于谱聚类的 BaaS 资源负载均衡调度算法

熊衍捷^a, 高 镇^b, 李 根^b, 杨晋生^a

(天津大学 a. 微电子学院; b. 电气自动化与信息工程学院, 天津 300072)

摘要: Fabric 作为超级账本的核心项目, 以其多通道的设计为用户提供更为隐私的交易空间, 为了解决基于分布式架构下的多通道资源负载均衡问题, 提出了基于 NJW 谱聚类的区块链即服务 (BaaS) 负载均衡调度算法 SC-channel。该算法将平台子节点的数量作为划分类簇数量的依据, 首先, 基于通道采用 peer 之间的 Jaccard 系数构造相似矩阵; 其次, 计算拉普拉斯矩阵, 求取前 k 个特征值和特征向量并将特征向量单位化; 最后, 用基于数量加权的 k-means 算法完成聚类。在 Kubernetes 平台上对这种方法进行验证, 并与采用经典 k-means 的 NJW 算法默认调度算法下的资源负载均衡度做了比较分析。结果表明, 采用基于谱聚类的 BaaS 资源负载均衡调度算法可提高资源利用的均衡程度, 增强了平台的可用性与可靠性。

关键词: Fabric; 谱聚类; 区块链即服务; Kubernetes

中图分类号: TP301.6

文献标志码: A

文章编号: 1000-582X(2021)11-040-08

BaaS resource load balancing scheduling algorithm based on spectral clustering

XIONG Yanjie^a, GAO Zhen^b, LI Gen^b, YANG Jinsheng^a

(a. School of Microelectronics; b. School of Electrical and Information Engineering,
Tianjin University, Tianjin 300072, P. R. China)

Abstract: As one of the core frameworks of the Hyperledger, Fabric provides users with private transaction space with its multi-channel design. In order to solve the problem of multi-channel resource load balancing based on distributed architecture, a Blockchain as a Service (BaaS) load balancing scheduling algorithm SC-channel based on NJW spectral clustering was proposed. The proposed algorithm took the number of platform sub-nodes as the basis for classifying the number of clusters. Firstly, based on channel the Jaccard coefficient between peer was used to construct the similarity matrix. Secondly, the Laplacian matrix was calculated to obtain the first k eigenvalues and eigenvectors, and the eigenvectors were unitized. Finally, the feature clustering was done using the classical weight-based k-means algorithm. The proposed algorithm was validated on the Kubernetes platform and its resource balance degree was compared with those of the NJW algorithm using the classic k-means and the default scheduling algorithm. Theoretical analysis and experimental results show that the BaaS resource load balancing scheduling algorithm based on

收稿日期: 2020-12-31

基金项目: 天津市自然科学基金资助项目(19JCYBJC15700)。

Supported by the Natural Science Foundation of Tianjin(19JCYBJC15700).

作者简介: 熊衍捷(1994—), 男, 硕士研究生, 主要研究方向区块链技术和云计算, (E-mail) xiongyj@tju.edu.cn。

通讯作者: 高镇, 男, 副教授, (E-mail) zgao@tju.edu.cn。

spectral clustering can improve the balance of resource utilization and enhance the usability and reliability of the platform.

Keywords: Fabric; spectral clustering; blockchain as a service; Kubernetes

近年来,区块链^[1-2]技术不断发展壮大,作为区块链与云计算结合的产物——区块链即服务,正改变着传统的云计算服务模式^[3],使区块链技术成为一种触手可及、开箱即用的工具服务于大众。超级账本^[4](Fabric)是 BaaS 的核心模块,提供基于联盟链的身份认证、交易背书及区块上链等功能。网络节点 peer 负责维护账本(Ledger),依据具体功能的不同,可分为提交节点(committer peer)和背书节点(endorser peer),Fabric 从 1.0 版本开始采用了多通道(channel)的设计,尽管有利于增强用户的隐私性和系统的扩展性,但是每一个 peer 均要处理来自不同通道的事务且保存一份相应的账本,将对有限的计算资源带来挑战并增大磁盘空间的存储压力。改善 BaaS 的性能已经变得愈加迫切。

基于 Kubernetes^[5]的云计算平台,才丽^[6]提出了面向 BaaS 的 Best-Balanced 加权评分模型解决资源负载均衡问题,Best-Balanced 算法从资源量负载平衡、资源量剩余量平衡和不同资源类型间负载平衡来衡量资源分配的平衡性。Medel 等^[7]通过基于 Petri 网的性能模型对 Kubernetes 容器的部署和终止开销进行分析,提出合理的 Pod 下容器配额规则,最后实现一种考虑容器资源争夺现象的调度算法。

基于多通道的设计,如果多个 peer 加入了同一通道并被调度至同一虚拟机,即使通道内的业务量不大,其任意时间点的负载均为每一个 peer 负载的总和,也可能造成虚拟机的负载过重,甚至发生单点故障,从而降低用户服务质量。针对上述问题,笔者提出了一种面向多通道的基于谱聚类(Spectral Clustering)的负载均衡调度算法 SC-channel,使用典型的 NJW^[8]谱聚类算法进行聚类分析,采用 Jaccard 相似系数依据对应的通道度量 peer 之间的相似度,使用基于数量加权的 k-means 算法做最后的特征聚类,并与平台默认提供的调度算法进行对比,实验结果证明了所提出算法能够增强系统的可靠性与可用性。

1 SC-channel 算法描述

谱聚类算法^[9]是机器学习领域中的一种无监督学习算法,基于图的分割理论划分样本集,以达到类内样本点距离最小而类间的样本点距离最大。相较于传统的聚类算法,如 k-means 聚类^[10]、层次聚类^[11]、基于密度的聚类(DBSCAN)^[12]等,能够在任意形状的样本空间上聚类且收敛于全局最优解。为了充分发挥谱聚类适应性强、收敛度高的优势,需要合理地选择具体谱聚类算法并依据问题空间选择度量样本间距离的方式。

1.1 算法可行性分析

如图 1 所示,Fabric 产生一笔交易的基本流程如下:

Step1 客户端创建一笔交易,该交易包括客户端的 ID、链码(chaincode)ID、提交交易自身的载体、时间戳、其他域客户端的签名,依据背书策略被发送到所选择的背书节点;

Step2 背书节点模拟交易并产生背书签名;

Step3 客户端收集到足够多的交易背书后,通过排序(ordering)服务广播该交易,如果客户端没有设法为交易收集背书,则放弃这个交易并稍后再试;

Step4 排序服务向各 peer 节点提交交易,在完成最终的确认后交易被提交上链。

图 1 中的①②③④对应于 step1、2、3、4。除步骤 3 外其余基本流程均需要 peer 的响应,因此 peer 在 Fabric 中承担着主要的计算任务。

如表 1 所示,该实验将 4 个 peer 以两两一组的方式分别放置于 2 台虚拟机(1G2 核)192.168.10.157 和 192.168.10.158 上,创建通道 channel1 和 channel2,给定每秒处理的交易量分别为 6 和 3 进行 2 组实验,每组实验重复 3 次,结果如图 2、图 3 所示。

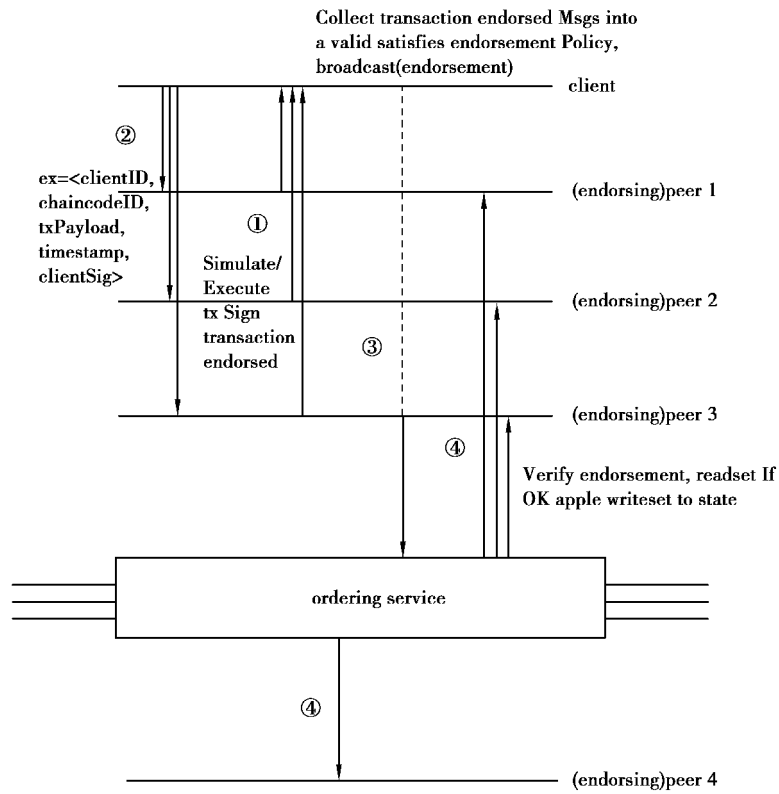


图 1 交易的基本流程

Fig. 1 The basic process of transaction

表 1 部署 peer 的两组实验

Table 1 Two experiments of deploying peers

peer	channel	expl-node	exp2-node
peer1	channell	192.168.10.157	192.168.10.157
peer2	channell	192.168.10.158	192.168.10.157
peer3	channel2	192.168.10.157	192.168.10.158
peer4	channel2	192.168.10.158	192.168.10.158

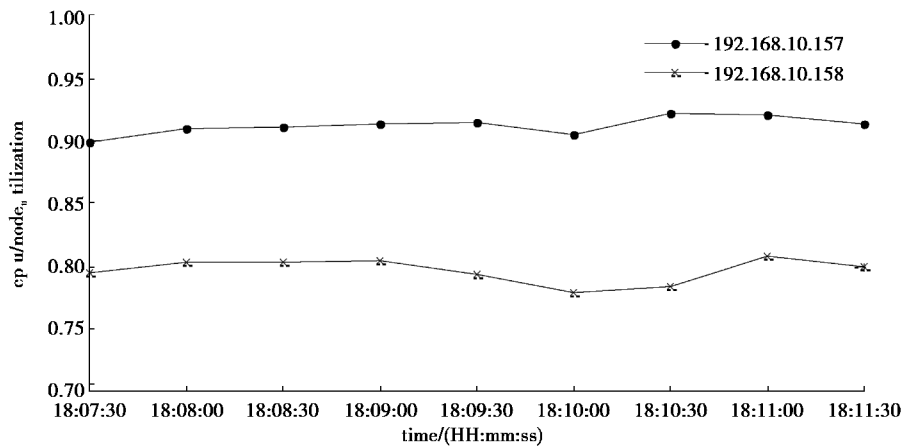


图 2 同一通道的 peer 在不同虚拟机的结果

Fig. 2 Results of peer in the same channel in different virtual machines

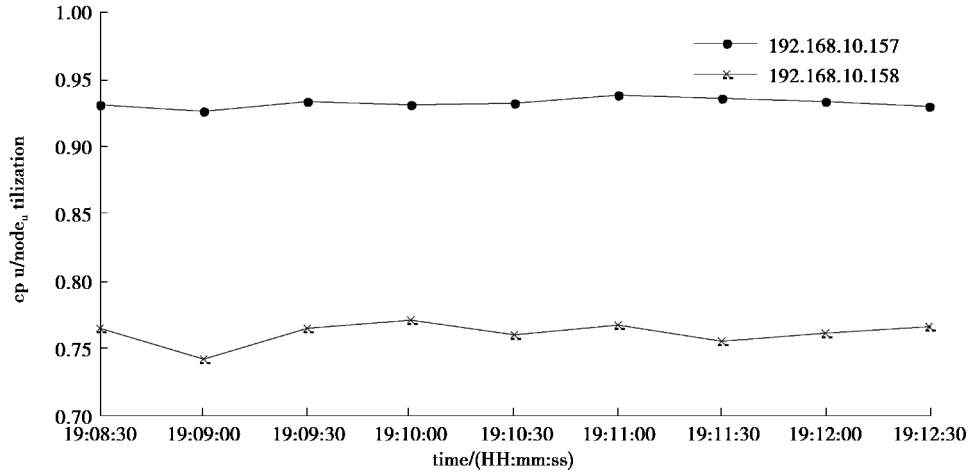


图 3 同一通道的 peer 在相同虚拟机的结果

Fig. 3 Results of peer in the same channel in the same virtual machine

如图 2 所示,将同一通道的 peer 放置于不同虚机上,通过对 peer 合理的分类,使各虚拟机 cpu 利用率更接近平台的平均 cpu 利用率,负载的均衡度更高,然而这仅考虑的是一种简单的场景,如果 peer 的数量较多,需要创建的通道数量也较多,那么如何确定分类原则将变得复杂。由于谱聚类算法不依赖于样本的形状,聚类效果优异,因此采用该算法对复杂场景下的 peer 做聚类,以此来提高平台的可用性。

1.2 算法思想描述

根据图划分准则的不同,经典的谱聚类算法可分为 PF 算法^[13]、SM 算法^[14]和 NJW 算法等,NJW 算法采用高斯径向核(Radial Basis Function, RBF)构造相似矩阵 \mathbf{W} ,即

$$w_{ij} = \exp\left(-\frac{d(s_i, s_j)}{2\sigma^2}\right). \quad (1)$$

能够直接得到聚类效果较好的样本集 k 路分割,因此选择 NJW 算法作为文中谱聚类算法的具体实现,该算法的主要步骤如下:

Step1 计算相似矩阵 \mathbf{W} ,获得度矩阵 \mathbf{D} ,得到拉普拉斯矩阵 \mathbf{L} ;

Step2 计算矩阵 \mathbf{L} 的前 k 个最大特征值所对应的特征向量 x_1, \dots, x_k ,构造矩阵 $\mathbf{X} = [x_1, \dots, x_k] \in \mathbb{R}^{n \times k}$;

Step3 将矩阵 \mathbf{X} 按行标准化为单位向量,得到矩阵 \mathbf{Y} , $y_{ij} = x_{ij} / \left(\sum_k x_{ik}^2\right)^{1/2}$;

Step4 使用 k-means 算法或其他经典算法按行对矩阵 \mathbf{Y} 聚类。

尽管 NJW 算法具有聚类效果好、多路分割的优势,但是簇类数目 C 和尺度参数 σ 需要人为选定,具有一定的不确定性。对于簇类数目 C 的选定,文献[15]提出了一种启发式确定类数的 DP-NJW 谱聚类算法,根据密度分布确定类中心和类数,Zeng 等^[16]通过使用模糊聚类测量谱聚类数目,Li 等^[17]基于传统的量子聚类,能够在不知道聚类数量的情况下找到任何形状的聚类,进一步缩短了计算时间,提高了计算效率;对于尺度参数 σ 的选择,Xie 等^[18]提出了局部标准偏差(Local Standard Deviation Spectral Clustering, SCSD)谱聚类算法改进 NJW 和 Self-Turning 算法^[19]对 σ 的选取,Mohanaval- li 等^[20]计算每个数据点 i 与其他点的距离,基于该点的度 D_i 使用四分位数自动确定 σ_i ,具有较好的聚类效果,文献[21]提出 SC_SD 和 SC_MD 两种谱聚类算法,其定义样本的标准差与样本到其余样本的距离的均值作为邻域半径,具备完全的自适应性。

将平台下具备计算能力的虚拟机数量作为簇类数目 C ,使得类之间隔离计算资源,能够较好解决人为因素带来的影响。相似矩阵 \mathbf{W} 的构造,由于 peer 样本集属于离散型、集合型数据,不再适用于采用 RBF 构造的相似矩阵,因此采用 Jaccard 相似系数度量样本间的距离。Jaccard 系数适用于表示有限样本集合的相似性与差异性,假定 x 和 y 分别具有 n 维特征(n 个通道), x_i 和 y_i 的取值为二元类型数值 $\{0, 1\}$,对于计算 x 与 y 的距离,Jaccard 系数使用 a, b, c, d 4 个参数来计算相似度,即

$$S_j(x, y) = \frac{a}{a + b + c}, \quad (2)$$

式中:1) a 表示样本 x 和 y 中满足 $x_i = y_i = 1$ 的数量,在 SC-channel 中表示 x 和 y 加入同一通道的数量。2) b 表示样本 x 和 y 中满足 $x_i = 1, y_i = 0$ 的数量,在 SC-channel 中表示 x 加入某通道而 y 未加入该通道的数量。3) c 表示样本 x 和 y 中满足 $x_i = 0, y_i = 1$ 的数量,在 SC-channel 中表示 x 未加入某通道而 y 加入该通道的数量。4) d 表示样本 x 和 y 中满足 $x_i = y_i = 0$ 的数量,在 SC-channel 中表示 x 和 y 均未加入同一通道的数量。

由式(2)可知,加入相同通道数量越多的 x 和 y 相似度越高,越趋向于分入同一个类,而据 1.1 节实验可知,加入不同通道的 x 和 y 处于同一虚拟机,即处于同一类簇,使负载均衡度提高,文中使用改写的 Jaccard 系数构造相似矩阵,即

$$S_{\text{SC-channel}}(x, y) = 1 - S_j(x, y) = \frac{b + c}{a + b + c}, \quad (3)$$

使用式(3)计算 peer 间的相似度将更适合文中的研究问题。SC-channel 算法伪代码如算法 1 所示。

算法 1 SC-channel 伪代码

输入:数据集 $\text{Peer} = \{\text{peer}_1, \text{peer}_2, \dots, \text{peer}_n\}$, 标签集 $\text{Channel} = \{\text{channel}_1, \text{channel}_2, \dots, \text{channel}_m\}$, 类别数 C 。

输出:聚类后的各簇 $H = \{h_1, h_2, \dots, h_C\}$, 按簇 binding 至 C 台虚拟机。

1) 遍历 peer 根据标签信息利用式(3)求出邻接相似矩阵 W ;

2) 计算度矩阵 D 和拉普拉斯矩阵 L , 其中, $D = \sum_{j=1}^n w_{ij}$, $L = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$;

3) 计算 L 前 C 个最大特征值对应的特征向量 v_1, v_2, \dots, v_C , 构造矩阵 $V = [v_1, v_2, \dots, v_C]$, 对矩阵 V 按行标准化为单位向量, 得到矩阵 Y , $y_{ij} = x_{ij} / \left(\sum_k x_{ik}^2 \right)^{1/2}$;

4) k-means 终止条件设置为聚类中心点不再变动或达到最大循环次数, 使用 k-means 算法按行对矩阵 Y 聚类成 C 类; 该结果易产生仅包含几个顶点的较小簇, 使机器的初始资源分配不均, 因此, 在 k-means 迭代过程中增加平衡因子 P 以提高各簇之间的均匀程度, 即检查分配的簇类样本数量是否大于 $\frac{n}{C}$, 若大于, 该簇类的目标函数同乘 P , 否则不变;

5) Kubernetes 调度器执行 binding 操作, 按聚类结果将 peer pod 调度至不同虚拟机。

2 SC-channel 实验

笔者在基于 Kubernetes v1.14.2 的云平台上用 5 台虚拟机进行了实验验证, 实验配置详情如表 2、表 3 所示, 通道创建详情如表 4 所示。使用 Python、Shell 混合编写 SC-channel 算法放于 Master 监听 peer 的调度, k-means 算法参考 sklearn.cluster 包改进。在 Master 机器使用 Fabric Nodejs SDK 向 BaaS 发起 invoke 合约调用, 部署 Heapster、Influxdb 对实验数据进行监视和收集, 取平衡因子 P 为 15, 为了校验资源调度算法的有效性, 采用资源利用率 V_i 的方差评价资源负载均衡度, 即

$$V_i = \alpha V_{\text{icpu}} + \beta V_{\text{iram}}, \quad (4)$$

$$S = n / \left[\sum_{i=1}^{i=n} (V_i - \bar{V})^2 \right]. \quad (5)$$

由于 BaaS 上 CPU 相比于内存变化波动较大, 式(4)中取 α 为 0.3, β 为 0.7。比较了 SC-channel 与仅仅采用经典的 k-means 的 NJW 算法 (NJW)、(LRP, Least Requested Priority)、SSP, Selector Spread Priority) 对 9 个 peer pod 的调度实验。

表 2 虚拟机配置详情

Table 2 Virtual machine configuration details

实验设备	型号/版本
CPU	intel i7-7700HQ 4 核/2 核
内存	DDR4 4GB/2GB
操作系统	CentOS Linux release 7.3.1611
Fabric	1.0
Docker	17.06.2-ce

表 3 集群网络拓扑

Table 3 Cluster network topology

IP 地址	节点角色
192.168.10.155	Master
192.168.10.157	Node
192.168.10.158	Node
192.168.10.160	Node
192.168.10.161	Node

表 4 通道的创建

Table 4 Creation of channel-peers

Peer	Channel
peer ₁	channel _{1,2,3}
peer ₂	channel _{2,3}
peer ₃	channel ₁
peer ₄	channel _{1,2,3}
peer ₅	channel _{2,3}
peer ₆	channel ₂
peer ₇	channel _{1,2,3}
peer ₈	channel _{2,3}
peer ₉	channel _{1,2,3}

如图 4 所示, channel₁ 与 channel₃ 始终保持较低吞吐率, channel₂ 在第 30 s 发起交易, 第 150 s 处结束, 逐步增大 TPS 量, 结果显示, 负载均衡度随着不断增长的交易量呈现出先下降后上升的态势, 当 channel₂ 具有小于等于 15 的吞吐量时, SC-channel 与其他 3 种调度算法下的负载均衡度总体相差不大, 随着 TPS 的不断提高, 二者负载均衡度拉开差距。这是由于负荷的提高, 平台下的一部分虚拟机已经达到了资源使用上限, 而那些计算资源充足的虚拟机因 peer 未加入始终无法释放潜在的能力。

统计每组实验的最高 S 值如表 5 所示。可以看出, 当通道接近空载, 默认调度算法 LRP、NJW 的均衡度更高, 且二者相差不大, 但随着处理量的增大, 均衡度值不断降低, 下降极为迅速, 将降低用户响应时间, 增大单点故障概率, 若 Pod 因资源不足而迁移将产生额外的资源损耗, 影响服务质量。随着 TPS 的提高, 使用经典 k-means 的 NJW 调度算法与 LRP、SSP 的均衡度差距不断缩小, 甚至在 TPS 为 40 时, 相比于 SSP 均衡度更高, 由于该算法产生了几个数量极少的簇, 导致相应的虚拟机始终处于空载的状态, 该算法更适应于庞大交易量的定向通道的业务场景, 使那些需要处理繁重业务的少数 peer 分布在一个虚拟机中。SC-channel 在 20~40 的 TPS 范围具有较高的均衡度, 随着 TPS 的提高, 均衡度表现得比较稳定。根据表 5 计算 4 种算法 NJW、LRP、SC-channel、SSP 下的平均 S 值分别为 88.42、72.78、62.60 和 83.94, 可见 SC-channel 负载均衡性能较好。

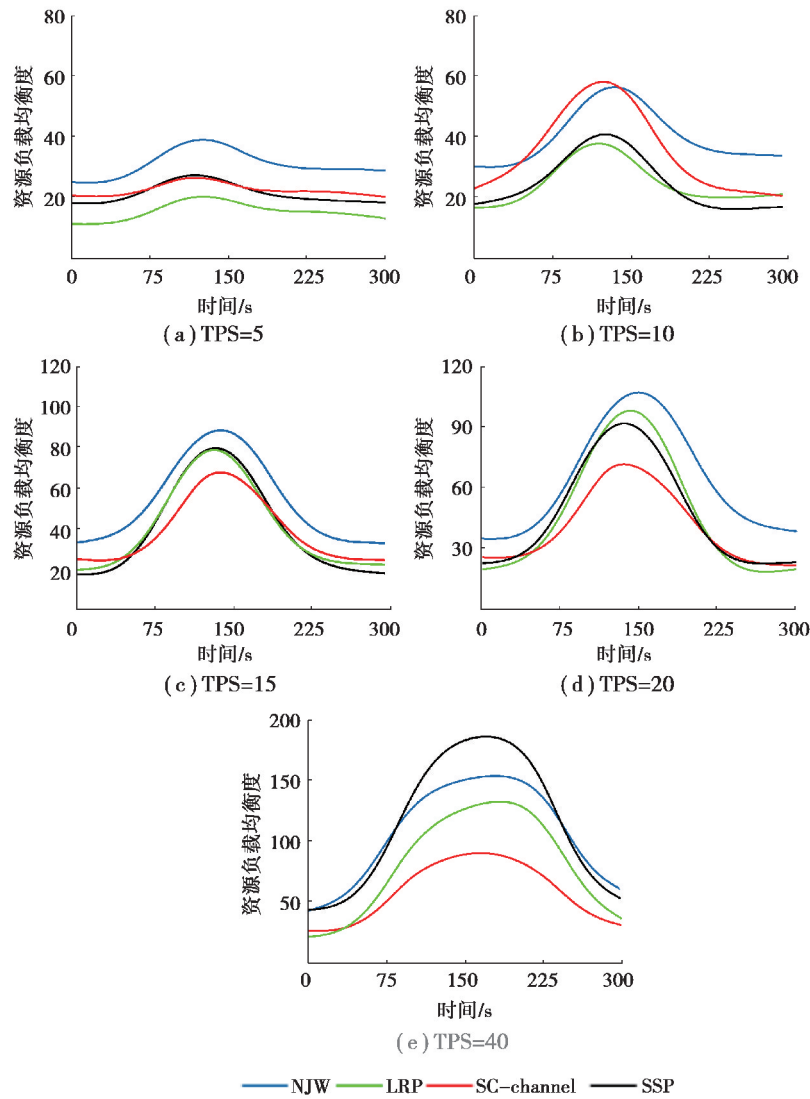


图 4 TPS 增大时 2 种算法下的负载均衡度

Fig. 4 Load balance degree under two algorithms when TPS increases

表 5 每组实验取最高值 S Table 5 The highest value of s taken in each run of experiments

调度算法	资源负载均衡度 S				
	5TPS	10TPS	15TPS	20TPS	40TPS
NJW	40.13	55.26	88.21	107.24	151.24
LRP	19.24	37.41	76.24	101.69	129.34
SC-channel	26.21	58.10	67.28	72.14	89.23
SSP	27.14	39.64	77.94	91.77	183.20

3 结 论

区块链技术以其去中心、去信任的优点成为当前社会的研究热点,其衍生的比特币、以太坊、超级账本等区块链项目正焕发生机,研究时下热门的 BaaS 平台,提出了基于 NJW 谱聚类的资源负载算法,该算法改进默认调度算法的不足,基于 peer 加入的通道作为调度考量,实验结果表明,利用该算法对 peer 进行调度使平台面对不同负载压力下的负载均衡度显著提高,增强了平台的可用性与可靠性。

参考文献:

- [1] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system [DB/OL]. (2006) [2019-5-26]. <https://bitcoin.org/bitcoin.pdf>.
- [2] 邵奇峰, 金澈清, 张召, 等. 区块链技术: 架构及进展[J]. 计算机学报, 2018, 41(5): 969-988.
Shao Q F, Jin C Q, Zhang Z, et al. Blockchain: architecture and research progress[J]. Chinese Journal of Computers, 2018, 41(5): 969-988.(in Chinese)
- [3] Blockchain As A Service [EB/OL]. (2017) [2019-6-28]. <https://it.toolbox.com/blogs/trevoreddolls/blockchain-as-a-service-042317>.
- [4] Hyperledger Whitepaper [EB/OL]. (2016) [2019-5-23]. <https://www.yumpu.com/xx/document/read/55615753/hyperledger-whitepaper>.
- [5] 龚正. Kubernetes 权威指南: 从 Docker 到 Kubernetes 实践全接触[M]. 北京: 电子工业出版社, 2017:12-14, 238-242.
Gong Z. Kubernetes definitive guide: from Docker to Kubernets practice[M]. Beijing: Publishing House of Electronics industry, 2017: 12-14, 238-242.(in Chinese)
- [6] 才丽. 面向 BaaS 平台的资源调度算法研究与实现[D]. 杭州: 浙江大学, 2018.
Cai L. The BaaS platform orientated research and implementation for scheduling algorithm[D]. Hangzhou: Zhejiang University, 2018. (in Chinese)
- [7] Medel V, Tolosana-Calasanz R, Bañares J Á, et al. Characterising resource management performance in Kubernetes [J]. Computers & Electrical Engineering, 2018, 68: 286-297.
- [8] Nga Y, Jordanm I, Weiss Y. On spectral clustering: analysis and an algorithm[C]// Proceedings of the 14th Advances in Neural Information Processing System, 2002:849-856.
- [9] 蔡晓妍, 戴冠中, 杨黎斌. 谱聚类算法综述[J]. 计算机科学, 2008, 35(7): 14-18.
Cai X Y, Dai G Z, Yang L B. Survey on spectral clustering algorithms[J]. Computer Science, 2008, 35(7): 14-18. (in Chinese)
- [10] 于佐军, 秦欢. 基于改进蜂群算法的 K-means 算法[J]. 控制与决策, 2018, 33(1): 181-185.
Yu Z J, Qin H. K-means algorithm based on improved artificial bee colony algorithm[J]. Control and Decision, 2018, 33(1): 181-185.(in Chinese)
- [11] Bouguettaya A, Yu Q, Liu X M, et al. Efficient agglomerative hierarchical clustering [J]. Expert Systems With Applications, 2015, 42(5): 2785-2797.
- [12] Tran T N, Drab K, Daszykowski M. Revised DBSCAN algorithm to cluster data with dense adjacent clusters [J]. Chemometrics and Intelligent Laboratory Systems, 2013, 120: 92-96.
- [13] Perona P, Freeman W. A factorization approach to grouping [C]// Computer Vision—ECCV'98. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998: 655-670.
- [14] Shi J B, Malik J. Normalized cuts and image segmentation [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000, 22(8): 888-905.
- [15] 陈俊芬, 张明, 何强. 基于启发式确定类数的 NJW 谱聚类算法[J]. 计算机科学, 2018, 45(S2): 474-479.
Chen J F, Zhang M, He Q. Heuristically determining cluster numbers based NJW spectral clustering algorithm [J]. Computer Science, 2018, 45(S2): 474-479.(in Chinese)
- [16] Zeng C P, Zhou A M, Zhang G X. Self-adaptive spectral cluster number detecting with particle swarm optimization algorithm [C] // 2016 IEEE Congress on Evolutionary Computation (CEC). July 24-29, 2016, Vancouver, BC, Canada. IEEE, 2016: 4607-4611.
- [17] Li Y Y, Wang Y, Wang Y Y, et al. Quantum clustering using kernel entropy component analysis [J]. Neurocomputing, 2016, 202: 36-48.
- [18] Xie J Y, Zhou Y, Ding L J. Local standard deviation spectral clustering [C]// 2018 IEEE International Conference on Big Data and Smart Computing (BigComp). January 15-17, 2018, Shanghai, China. IEEE, 2018: 242-250.
- [19] Zelnik-Manor L, Perona P. Spectral kernel methods for clustering [C] // Advances in Neural Information Processing Systems 14. Cambridge MA: The MIT Press, 2002.
- [20] Mohanavalli S, Jaisakthi S M, Aravindan C. Automatic scale parameters in affinity matrix construction for improved spectral clustering [J]. International Journal of Pattern Recognition and Artificial Intelligence, 2016, 30(10): 1650023.
- [21] 谢娟英, 丁丽娟. 完全自适应的谱聚类算法[J]. 电子学报, 2019, 47(5): 1000-1008.
Xie J Y, Ding L J. The true self-adaptive spectral clustering algorithms [J]. Acta Electronica Sinica, 2019, 47(5): 1000-1008.(in Chinese)