DOI:10.20033/j.1003-7241.(2025)09-0138-05

基于Kubernetes调度算法的动态负载均衡方法研究

贺敬伟，程伟华，张世杰
（江苏电力信息技术有限公司，江苏 南京 210024）

摘　要：为了降低动态负载均衡的失衡度，在动态负载均衡中识别负载信道状态时，充分考虑链路能耗对传输信道的影响，基于Kubernetes调度算法对动态负载均衡方法进行研究。利用Kubernetes调度算法研究动态资源的调度机制，在调度机制下结合链路能耗对通信业务分类；识别负载信道状态，并以此为依据设计动态负载均衡模式；通过对通信网络中的负载判决进行判定，从而实现动态负载均衡的目的。在实验论证中，对所提方法的综合性能进行了验证。实验结果证明，设计方法的负载失衡度始终较低，最高为4%，远低于对比方法，说明设计方法可以很好地均衡动态负载。

关键词：Kubernetes调度算法；动态负载均衡；负载失衡度；信道状态识别；动态负载均衡模式
中图分类号：TN929.5　　**文献标识码：**A　　**文章编号：**1003-7241(2025)09-0138-05

Research on Dynamic Load Balancing Method Based on Kubernetes Scheduling Algorithm

HE Jingwei, CHENG Weihua, ZHANG Shijie
(Jiangsu Electric Power Information Technology Co., Ltd., Nanjing 210024, China)

Abstract: In order to reduce the imbalance degree of dynamic load balancing, the influence of link energy consumption on transmission channel is fully considered when identifying the load channel state in dynamic load balancing, and the dynamic load balancing method is studied based on Kubernetes scheduling algorithm. The scheduling mechanism of dynamic resources is studied by using Kubernetes scheduling algorithm, and the communication services are classified according to the link energy consumption under the scheduling mechanism. The load channel state is identified and the dynamic load balancing mode is designed based on it. By determining the load decision in the communication network, the purpose of dynamic load balancing is realized. In the experimental demonstration, it verifies the applicability of the design method. The experimental results show that the load imbalance degree of the design method is always low, the highest is 4%, which is much lower than that of the comparison method, indicating that the design method can balance the dynamic load well.

Keywords: Kubernetes scheduling algorithm; dynamic load balancing; load unbalance; channel state identification; dynamic load balancing mode

0　引言

网络负载均衡问题是大规模资源平台中较为重要的问题，一种优秀的负载均衡方法，不仅可以为平台带来计算性能上的提升，还可以提高平台的稳定性[1]。

当前存在大量学术研究来解决动态负载失衡问题。曲乾聪等[2]基于负载反馈研究分布式数字集群动态负载均衡方法。建立参与MCPTT服务器的负载监控机制和相应指标，运用加权轮询算法对用户进行分配，求得复合负载参数，按照负载指标的反馈结果更新权重值，由此实现动态负载的均衡。该方法对于用户请求的响应延迟较低，但是动态负载的失衡度仍然有降低空间。李娟等[3]以物联网技术为基础，对异构集群动态负载均衡进行研究。利用物联网技术中的传感技术构建传输模型；利用动态加权法配置输出信道，实现对信道的特征分解；运用噪声干扰抑制方法对通信信道的干扰进行抑制，进行模糊度均衡配置和空间均衡调度，实现对动态负载的均衡处理。该方法能够有效降低输出信号的误码率，但是对于失衡度的控制仍然不够稳定。

为了解决动态负载均衡过程中失衡度较高的问题，本文提出了基于Kubernetes调度算法的动态负载均衡方法，充分考虑到负载容量、集群规模以及链路能耗等因素，制定资源调度机制，进而实现动态负载均衡，增强网络动态负载均衡系统的整体性能。

1　动态负载均衡方法设计

1.1　动态资源调度机制的研究

Kubernetes动态负载均衡的核心在于让Pod做重新调度。重调度的主要器件是Kubernetes调度器，其负责判断待调度集合中是否存在优先级资源Pod，并根据调度机

*基金项目：国家电网有限公司科技项目资助(1200/2020-02003B)
收稿日期：2023-12-29

制将资源与集群系统中的网络节点进行特殊绑定[4]。Kubernetes调度器框架如图1所示。

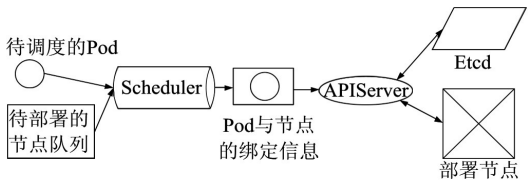

图1 Kubernetes调度器框架图

Kubernetes调度器的主要作用是通过管理节点管理通信网络中的集群系统，每个Kubernetes集群系统对应一个Master控制节点和被管理容器Node[5]。当控制节点对管理容器发出调度命令时，Kubernetes会判定该调度节点是否为优先级资源。在集群系统中，可以利用虚拟机和物理机作为管理容器的宿主，确保各节点能够并行运行包括Kubelet、Kube-Proxy以及Docker等在内的关键服务进程。同时，每个容器均有能力承载多个待调度的资源节点Pod，鉴于Pod是调度器管理的最基本单元。值得注意的是，一个Pod内部也可以整合多个管理容器，实现资源的灵活配置。在Master节点上，运行着四大核心服务进程：Kube-APIServer、etcd、Scheduler（调度器）及Kube-Controller-Manager。这些组件在控制节点上协同运作，共同负责整个集群的管理与调度[6]。

在Kubernetes所管理的容器集群环境中，如果Pod的配置文件通过NodeSelector明确指定了节点标签，或者在Pod创建时直接采用了PodFitsHost方式指定了具体的节点名称，那么这一绑定信息将会被记录到Etcd中，并且Pod会直接在指定的节点上被创建。这一过程中，Pod将不再经过Kubernetes调度器的筛选与评估环节。Pod的调度流程如图2所示。

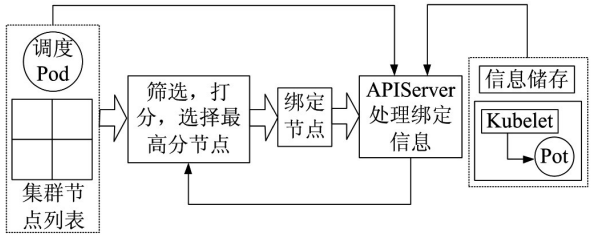

图2 Pod调度流程图

根据图2所示，Kubernetes调度器的核心调度流程主要包括以下步骤：

Kubernetes调度器会检查被管理节点和运行服务器上的状态，以识别是否存在尚未绑定的待调度Pod信息。一旦发现有待调度的Pod，则立即进入下一步处理；若未检测到待调度Pod，则进入定时检测模式以持续监控状态变化。

主机筛选阶段，调度器会排除那些不符合调度策略的主机。例如，如果Pod指定了所需的端口，而该端口在目标主机上已被占用，则该主机将被直接排除在候选列表之外[7]。

主机评分阶段，对于先前筛选出的满足条件的主机，调度器会根据一个预先设定的评分标准来给予它们相应的分数。这个评分标准在本文中是依据节点资源的优先级和均衡性原则来设计的，同时也提供了灵活性，使得调度器的评分逻辑可以根据实际需求进行自定义设置。如果一个新的调度节点被匹配到一个待调度信息，则这个节点上资源的优先级为该节点未匹配到资源的空间与总空间的比值，计算方法为

$$S_{cpu}=\frac{T_{cpu}-S_{usedu}-P_{requ}}{T_{cpu}}$$
$$S_{mem}=\frac{T_{mem}-S_{usedm}-P_{reqm}}{T_{mem}} \quad (1)$$
$$G=\frac{S_{cpu}+S_{mem}}{2}$$

式中，T_{cpu}和T_{mem}分别表示当前节点的CPU和内存总量；S_{usedu}和S_{usedm}分别表示当前节点已部署的Pod和其他应用消耗的CPU和内存总量；P_{requ}和P_{reqm}分别表示当前待调度Pod请求的CPU和内存用量；S_{cpu}和S_{mem}分别表示节点空闲的CPU和内存量占比；G表示节点资源优先级，比值越大，该节点的分数越高[8]。

选择最优主机：调度器会选择评分最高的主机作为Pod的部署目标，并将相关信息存储在Etcd中，随后在该主机上创建Pod。此机制旨在优化Pod的部署，避免将Pod放置在资源分配不均衡的节点上，从而防止某些节点因某一类资源耗尽而其它资源过剩的情况。节点的资源均衡性越高，被选中的可能性就越大，具体评估方式为

$$B=1-abs\left(S_{cpu}-S_{mem}\right) \quad (2)$$

节点总得分计算方法为

$$Score=p\left(G+B\right) \quad (3)$$

式中，G和B分别代表由公式(2)和(3)计算得出的节点得分，而$Score$代表节点的总得分。若检测到某节点的指定端口Port已被占用，则通过p=-1标记该节点为不可用，并赋予其负得分；反之，若端口未被占用，则p=1表示该节点处于可用状态。

至此，利用Kubernetes调度器完成了对于通信系统中动态资源调度机制的研究[9]。根据图2所示，该调度机制旨在确保Pod能够运行于性能最佳的节点上，从而优化其服务质量。这一策略为后续系统中实现动态负载均衡奠定了坚实的基础。

1.2 识别动态负载信道状态

通信网络中，负载的主要功能之一就是资源传输和

信息共享。因此，基于资源调度机制，对网络中的传输业务进行分类，并考虑链路能耗对负载信道的影响，对动态负载的平衡问题设计最优方案[10]。通信业务调用参数及内容如表1所示。

表1 通信业务调用参数及内容

Namespace	通信业务调用参数	内容
UTS	CLONE NEWUTS	主机名与域名
IPC	CLONE NEWIPC	信号量、消息队列和共享内存
PID	CLONE NEWPID	进程编号
Network	CLONE NEWNET	网络设备、网络栈、端口等
Mount	CLONE NEWNS	挂载点(文件系统)
User	CLONE NEWUSER	用户和用户组

一般情况下，网络节点之间的信息输送是通过负载对应的信道来实现的，并根据资源调度机制确定负载信道状态。由于通信系统中的无线信道具有时变性，信道在一定范围内的状态会发生改变，因此需要确定信道在接收和传输数据期间的状态转移规律[11]。衡量通信信道状态的指标有信噪比和门限值。在通信系统中，利用调度编码方式计算信道状态δ的发证概率，为

$$P(\delta)=\int_0^u l\left(\frac{1}{\delta}\right) \tag{4}$$

式中，δ表示信道接收和传输资源时的状态；l表示当前资源所占用的信道；u表示资源特征融合参数。

在此基础上，可得概率密度函数，为

$$G(\alpha)=\frac{1}{=\atop\alpha}\exp\left(-\frac{\alpha}{=\atop\alpha}\right)^{-1}\times P(\delta) \tag{5}$$

式中，$\frac{1}{=\atop\alpha}$表示通信系统接收数据时的嵌入层网络参数；α表示通信系统传输数据时的嵌入层网络参数；$\bar{\alpha}$表示调整通信系统嵌入层网络参数变化情况；$P(\delta)$表示信道状态的发证概率。

假设通信系统的无线信道在状态转移期间与相邻的信道没有产生交集，因此，可以得到信道状态发生小范围转移的概率为

$$R_\delta=G(\alpha)\times\sqrt{\frac{2\pi\delta}{=\atop\alpha}}E \tag{6}$$

式中，$G(\alpha)$表示信道状态δ的概率密度函数；E表示通信网络最大多普勒频移。

通信系统信道切换的方式主要包括主动切换和被动切换。在相邻的两个网络基站之间的覆盖切换为主动切换[12]。在切换过程中，假定在同一时间内系统输入的数据量与输出的数据量成正比关系，因此，数据包传输为

$$L=R_\delta\times\frac{(H_e)^{\phi_{\text{in,out}}}}{\phi_{\text{in,out}}}\exp(-H_e) \tag{7}$$

式中，R_δ表示信道状态δ发生转移的概率；H表示数据分流状态；$\phi_{\text{in,out}}$表示同一时间内输入与输出的数据总量；e表示数据量传输时刻。

由此可以推导出在传输时刻e+1时通信网络中数据总量为

$$L_{e+1}=\frac{H_{e+1}}{\min\left\{\phi_{\text{in,out}}+\beta_{e+1},D\right\}} \tag{8}$$

式中，β表示选路效率；D表示信道资源数量。

为了保证通信系统的数据传输效率，通常情况下选择成本最低的无线信道来满足多个类型的信道切换。随着切换进程的增加，信道的传输速率会相应减缓。因此，有必要在两个相邻的网络基站间执行任务切换。当单一的切换进程占用负载信道时，资源传输的速率为

$$g_\varepsilon^0=r\log_2\left(1+\frac{\varepsilon}{\gamma^2}\right)^{L_{e+1}} \tag{9}$$

式中，r表示单一切换进程；ε表示资源在传输过程中的时延；γ表示信号传输过程中的接收时延；L_{e+1}表示网络节点数据量。

针对时延问题，可以通过传输筛选策略对传输的移动设备进行区域迁移，计算方法为

$$Q=\begin{cases}t\to F,\varepsilon\leqslant\xi\\ t\to F,\gamma>\xi\end{cases} \tag{10}$$

式中，t表示优化后的任务切换结果；F表示任务切换群组；ξ表示某设备上的信号传输速率。

基于上述计算流程，可以通过分析动态负载信道的状态，来确定负载信道在数据传输期间发生范围转移的概率。为了有效减少通信网络中的链路能耗，依据计算结果，对通信网络的基站切换策略进行优化配置。

1.3 设计动态负载均衡模式

在明确负载信道状态的前提下，评估资源信息的交换模式。若资源处于交换模式，则通信系统中所有负载端的负载值将维持在设定的门限值范围内。为评估网络节点间的协作可能性，需要收集用户终端的数据、网络基站的相关参数以及负载端的负载数值，并基于这些信息来进行计算。

$$S=\frac{q\times g_\varepsilon^0}{\left\{V|V<V_{\min}\right\}} \tag{11}$$

式中，q表示通信网络的空间维度；g_ε^0表示资源输入速率；V表示系统传输信号的时延；$V_{\min}$表示通信网络可接受的最短时延。

通信设备接收到的信号强度为

$$V=\frac{1}{w}-k \tag{12}$$

式中，w表示通信系统的资源输出速率；k表示用户与通信系统传输资源时的链路能耗。在此基础上，得出通信

网络节点到网络基站的位置坐标为

$$d=\frac{\sqrt{\left(x_0-x_1\right)^2+\left(y_0-y_1\right)^2}}{S\times\sqrt{\eta}} \tag{13}$$

式中，x_0表示资源网络基站参数；y_0表示用户使用终端参数；x_1和y_1表示网络基站的位置坐标；S表示网络节点的协作概率；η表示负载值。通过评估系统内的负载判决空间，最终达成负载均衡决策的目标。

2 实验论证

本文设计仿真实验来证明所提的基于Kubernetes调度算法的动态负载均衡方法在实际应用中的综合性能，并运用对照实验，对得到的实验结果进行对比分析。

2.1 实验说明

本次实验共计选用25台性能参数一致的物理机作为通信网络中的主机节点：CPU处理能力（MIPS）{1 000，1 800，2 600，3 000}、内存（G）{1，2，4，8}、带宽（Mb/s）{500，700，1 000}；在以上25台物理机的主节点分别配置5个虚拟计算从节点：GPU处理能力（MIPS）{200，500，1 000，1 500，2 500}、内存（G）{0.5，1，2，3}、带宽（Mb/s）{100，2000，500}。仿真构建分布式云计算网络环境，并模拟不同云任务类型、不同规模服务数据驱动下的通信系统运行情况。实验场景中的网络基站在通信网络中的分布均服从正态分布，并在此基础上，将通信系统中基站发射的信号扩散范围设为200 m~500 m，保证信号强度；信号传输功率为42 dBm-46 dBm；将系统的背景噪声功率控制在-150 dBm/Hz以下；将系统的动态负载周期设为100 ms。在实验组中引入Kubernetes调度算法时，需配置一个如图3所示的个性化调度器，并在其中嵌入特定的调度策略。

```
apiversion:kubescheduler，config，k8s，io/v1a
lpha1
kind:kubeScheduterConfiguration
schedulerName:mY-scheduler
algorithmSource:
  policy:
    file:
      path:policy，yaml
leaderEle}tion:
  leaderElect:true
  lockObjectName:mY-scheduter
lockobjectNamespace:Kube-ystem
```

图3 调度配置文件

2.2 实验准备

实验环境是基于四台搭载麒麟操作系统的飞腾2000+（CFT-2000+）国产服务器部署的Kubernetes 1.14.2集群系统。其中，一台服务器被配置为Master主节点，其余三台则作为Node从节点，共同构成一个简化集群用于测试。主节点的IP地址为192.168.2.120。利用Kubernetes调度算法对集群系统进行模拟，每个进程对应着集群系统中的一个节点，每个进程拥有独立的线程池，线程与主服务器之间以RMI方式进行消息传递。

实验中所需全部软件均需通过源代码编译获得，或寻找对应Arm架构的二进制可执行文件。本实验选择编译Kubernetes源代码，并利用编译生成的各组件二进制文件进行安装。Kubernetes集群构建完成后，即可在集群上查看所有活跃组件的详细信息和运行状态。Kubernetes集群运行的逻辑流程示意如图4所示。

```
Ensure:w
For s=0,1,...s
Optionl(Random Sampling): Sampling with
replacement is done n times in the data set,
divided into K partial data sets.
Option2(Global Replacement): Random
replacement data set, divided Into K local data
sets.
Option3(Local Replacement): Working nodes
randomly replace local data sets.
For t=0,1,…,s
For k=1,…， K in parallel do Read the current
model
Randomly extract small batches of data from local
data sets(t)
Calculate the stochastic gradient {t}.(R)TT on Dk
iEDk (t)
end for
update global parameter
end for
```

图4 Kubernetes集群运行伪代码

表2列出了本实验环境中服务器上运行的Kubernetes集群核心组件的详细描述。

表2 Kubernetes集群相关组件描述

软件名称	功能描述
Docker18.06	应用运行环境
API Server-v1.14.2	提供资源对象增、删、改、查的接口
Scheduler-v1.14.2	负责Pod的调度
Controller Manager-v1.14.2	负责集群中所有资源对象的管理
Kubelet-v 1.14.2	负责Node节点资源的使用
Heaspter-v1.4.2-arm64	收集整个集群中资源使用数据
InfluxDB-v 1.3.3-arm64	用来存储历史资源的使用数据
Grafana-v4.4.3-arm64	用来展示资源的使用情况的界面
Flannel-v0.7.1-linux-arm64	负责解决跨主机网络连接
Kubectl-v 1.14.2	供用户使用的命令行工具
Kube-Proxy-v 1.14.2	负责具体的负载均衡
Etcd-v3.3.10-linux-arm64	用于存储各个Node节点和Pod信息

2.3 负载失衡度实验分析

为了避免实验结果的偶然性，采用文献[2]基于负载反馈的动态负载均衡方法（方法1）和文献[3]基于物联网技术的动态负载均衡方法（方法2）与本文设计的基于Kubernetes调度算法的动态负载均衡方法进行对比实验，统计三种方法在通信系统中的动态负载失衡度，统计及对比结果如图5所示。

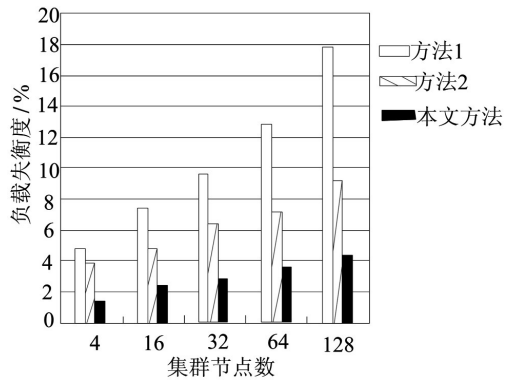

图5 动态负载失衡度对比结果

图5的对比结果表明，随着集群系统规模的不断增大，三种方法的负载失衡度均呈现出增大趋势。方法1由于未充分考虑集群系统本身存在的性能差异，在系统规模扩大时，其负载失衡度会明显升高，当集群节点数为128时，负载失衡度已达到了18%。方法2虽然考虑了系统性能的异同，但该方法是根据服务器能够承受最大负载能量来赋予访问页面权重值，忽略了集群负载率分布情况，导致负载均衡效果不是很理想，当集群节点数为128时，负载失衡度最高，为9%左右。本文方法所采用的基于Kubernetes调度算法会根据集群规模和负载率在通信系统中的分布对动态资源进行调度，当集群节点数为128时，负载失衡度达到最高值，此时其数值为4%，明显低于两种对比方法，说明本文方法取得了较好的负载均衡效果。

3 结束语

本文研究提出了一种基于Kubernetes调度算法的动态负载均衡方法，对集群容量、内存以及网络资源负载分布情况等进行多角度考虑，设计了资源调度机制，对网络节点的各类资源进行均衡利用，实现负载均衡的目的。该研究结果表明，随着集群系统节点数规模的不断增大，负载失衡度也会呈现出上升趋势，但本文方法下的负载失衡度最高为4%，明显优于对比方法，说明本文方法对于合理利用集群资源具有一定的作用，具备一定的实际应用意义。

参考文献：

[1] 付元光，刘鹏，李瑞，等. 蒙特卡罗临界计算粒子并行动态负载均衡调权算法研究[J]. 原子能科学技术，2022，56(2)：316-325.

[2] 曲乾聪，王俊. 基于负载反馈的分布式数字集群动态负载均衡算法[J]. 计算机应用研究，2022，39(2)：526-530，542.

[3] 李娟. 基于物联网技术的异构集群动态负载均衡算法[J]. 计算机与现代化，2021(4)：104-108.

[4] 宋锦华，马传琦. 基于移动边缘计算的舰船通信网络动态负载均衡方法[J]. 舰船科学技术，2021，43(24)：115-117.

[5] 刘玉鑫，程明，李泽华，等. 高压柜操作机器人协调操作感知控制方法[J]. 自动化技术与应用，2023，42(11)：49-52.

[6] 谭双杰，林宝军，刘迎春，等. 基于机器学习的分布式星载RTs系统负载调度算法[J]. 计算机科学，2022，49(2)：336-341.

[7] 周磊，孟利民，周立鹏，等. 基于二部图最大匹配的动态负载均衡算法[J]. 高技术通讯，2020，30(8)：798-804.

[8] 平凡，陈莉君. 基于Kubernetes的动态负载均衡机制研究与设计[J]. 计算机与数字工程，2020，48(1)：141-146.

[9] 刘毅，李凯心，李国燕，等. 基于SDN的动态负载均衡策略[J]. 计算机应用研究，2020，37(10)：3147-3152.

[10] 吴俊鹏，刘晓东. 一种基于集群的动态负载均衡算法研究[J]. 电子设计工程，2021，29(16)：75-78.

[11] 姚泽玮，林嘉雯，胡俊钦，等. 基于PSO-GA的多边缘负载均衡方法[J]. 计算机科学，2021，48(S2)：456-463.

[12] 刘凤元，王保根，江帆，等. 基于云计算的水电站数据安全存储系统[J]. 自动化技术与应用，2023，42(3)：97-100.

作者简介：贺敬伟(1982—)，男，本科，工程师，研究方向：云计算，企业信息化。

(上接第118页)

偿与信号调理设计与测试[J]. 计算机测量与控制，2021，29(2)：256-261，266.

[11] LI T，XUE H，WANG W. A high-pressure sensor with high linearity with s-shaped piezoresistors[J]. IEEE Sensors Journal，2022，23(2)：1052-1059.

[12] 张旭乐，陈刚，王群锋，等. 基于数字量输出的直流电压测量装置频率传变特性研究及误差分析[J]. 计量与测试技术，2023，50(3)：25-29.

[13] 张宇，王兰炜，胡哲. 地电阻率观测中两种电极接地电阻测量方法及其对比研究[J]. 地震学报，2022，44(6)：12.

[14] 魏振忠，司倡如，李振. 基于PSCAD的LNG接收站二次系统耐雷水平仿真分析方法[J]. 自动化技术与应用，2023，42(6)：160-164.

作者简介：廖路(1981—)，男，本科，工程师，研究方向：雷电防护技术。

通信作者：许伟(1974—)，男，硕士，正研级高工，研究方向：雷电防护技术。