基于 kube-vip 的高可用 Kubernetes 云平台实现

王湘渝[1] 曹 康[2]
WANG Xiangyu CAO Kang

摘 要 针对传统 HAProxy+Keepalived 高可用方案在 Kubernetes 集群部署中存在的架构冗余、配置复杂及资源消耗高等问题，文章设计并实现了一种基于 kube-vip 的云原生高可用架构。该方案通过深度集成 Kubernetes 控制平面，创新性地采用 Operator 模式动态感知服务状态变化，实现负载均衡规则实时更新；通过优化轻量级 Raft 选举协议，将领导者切换耗时压缩至毫秒级；并首创 ARP/BGP 双模混合网络架构，支持跨数据中心流量智能调度。在工程实践中，基于 kubeadm 部署 4 节点集群（3 主 1 从），通过 kube-vip 纯软件方式实现虚拟 IP（VIP）的故障自愈功能，将配置流程简化 60%。

关键词 kube-vip；高可用；云原生；容器；Kubernetes

doi：10.3969/j.issn.1672-9528.2026.04.038

0 引言

云计算技术的应用正从资源虚拟化向服务容器化加速转型。Kubernetes 作为容器编排的事实标准，得到了广泛的应用。但如果采用单 Master 节点的 Kubernetes 集群编排和管理容器，可能会出现因 Master 单节点故障，造成整个集群无法正常使用的情况，因此进行高可用性设计，能提高 Kubernetes 企业级应用的稳定性和连续性。根据 Gartner 2023 年报告，在未来两年内全球 75% 的企业将采用 Kubernetes 作为核心应用平台，其中金融、医疗等行业对高可用性的需求尤为迫切。

传统高可用方案（如 HAProxy+Keepalived）虽已在生产中使用，但一直存在架构冗余、运维复杂、跨云兼容性及资源效率方面的瓶颈问题，难以满足云原生场景的动态需求。例如，某金融机构采用 HAProxy+Keepalived 部署的 Kubernetes 集群因配置错误导致服务中断，平均恢复时间（MTTR）长达 15 min。

kube-vip 通过云原生集成了 Kubernetes 控制平面，可实现服务状态的实时同步与故障自愈，为关键业务提供“零感知”高可用保障，具备广阔的应用前景。本文基于 kube-vip 这一轻量级工具，提出一种深度集成 Kubernetes API 的高可用架构，通过理论模型构建与实践验证，解决了传统方案的局限性，为云原生基础设施的应用提供了新思路。[1]

1. 湖南科技职业学院软件学院 湖南长沙 410118
2. 湖南生物职业技术职业学院经贸学院 湖南长沙 410127
[基金项目] 湖南省职业院校教育教学改革研究项目“高职院校云计算技术应用专业‘岗课赛证’综合育人的路径探索”（ZJGB2022035）

1 问题描述

在 Kubernetes 集群中，Master 节点是核心控制节点，该节点部署了 API server、etcd、Controller Manager 等软件。其中 API server 作为集群的通信枢纽，负责接收所有 REST 操作请求并验证其合法性。

若该节点发现故障，如硬件损坏、网络隔离或进程崩溃，将直接导致以下严重后果：（1）集群管理瘫痪：kubectl 命令无法执行，新 Pod 调度停止，服务扩缩容失效；（2）数据一致性风险：etcd 若未组成集群，持久化数据可能丢失；（3）业务连续性中断：运行中的 Pod 虽可继续服务，但无法监控或修复异常实例 [2]。

高可用 Kubernetes 集群通过多 Master 节点冗余架构配备自动故障迁移功能：（1）实时检测：基于健康检查机制（如 kubelet NodeStatus）秒级感知节点故障；（2）快速转移：控制平面组件通过分布式共识算法（如 Raft）在百毫秒内选举新 Leader；（3）服务无缝接替：虚拟 IP（VIP）漂移至健康节点，使 API server 服务 IP 保持不变，对外屏蔽故障。

该机制将故障恢复过程自动化，减少对管理员干预的依赖，确保集群 SLA ≥ 99.95%。

2 Kubernetes 高可用集群概念

Kubernetes API server 提供了 Kubernetes 各类资源对象（Pod、RC、server 等）的增删改查等 HTTP Rest 接口，是整个系统的数据总线和数据中心。

2.1 Kubernetes API server 的功能

（1）提供了集群管理的 Reset API 接口（包括认证授权、数据校验以及集群状态变更）。

（2）提供其他模块之间的数据交互和通信的枢纽（其他模块通过 API server 查询或修改数据，只有 API server 才能直接操作 etcd）。

（3）是资源配额控制器的入口。

（4）拥有完善的集群安全机制。

API server 是用户和 Kubernetes 集群交互的入口，封装了核心对象的增删改查操作，提供了 RESTFul 风格的 API 接口，通过 etcd 实现持久化并维护对象的一致性。在整个 Kubernetes 集群中，API server 服务至关重要，一旦宕机，整个 Kubernetes 平台将无法使用，所以保障企业高可用是运维必备的工作。

2.2 部署 Kubernetes 高可用集群的原因

（1）容错性

高可用集群可以确保即使某个控制节点或组件发生故障，集群仍然可以继续正常运行。其方法是在多个控制节点上部署 API server、Controller Manager 和 Scheduler 等。

（2）负载均衡

Kubernetes 高可用集群会在多个节点上部署相同组件，并通过负载均衡来分发流量，从而提高性能并减少单点故障的风险。

（3）可扩展性

高可用集群可以更轻松地进行扩展，满足业务不断增长的需求，其通过添加更多的节点或扩展现有节点的资源，增加集群的容量和性能。

（4）可靠性和稳定性

通过使用冗余的节点和组件，高可用集群可以提供更高的可靠性和稳定性，即使在节点出现故障或维护期间，高可用集群也可以保持可用性。

（5）自动恢复能力

2.3 Kubernetes 高可用集群的工作原理

多 Master 节点的 Kubernetes 高可用集群的架构如图 1 所示。

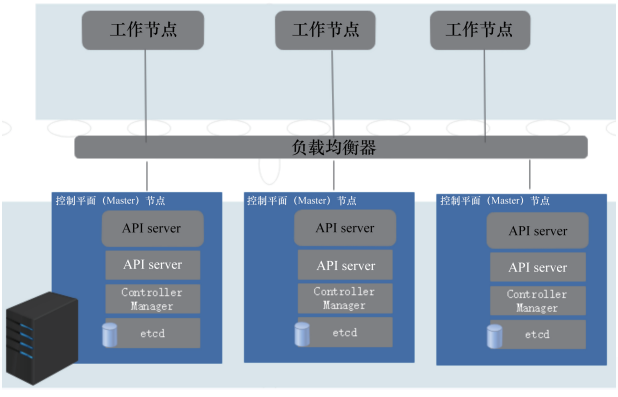

图 1 多 master 节点的 Kubernetes 高可用集群的架构

在 Kubernetes 高可用集群中，运行了 3 个 Master 节点，每个 Master 节点都会运行 API server、Controller Manager 和 Scheduler 等控制平面组件。当某个 Master 节点出现故障时，其他的 Master 节点仍然可以继续提供服务，整个集群的管理和控制工作不会受到影响。

（1）负载均衡器

负载均衡器服务将请求调度到多个 Master 节点上的 API server，确保所有发生 API server 的请求都能被成功地路由到一个可用的 Master 节点。

（2）etcd 集群

Kubernetes 使用 etcd 作为集群的数据存储。当实现 Kubernetes 的高可用时，etcd 部署在每个 Master 节点上，形成数据库集群并进行数据实时同步。若某个节点崩溃，则数据可以从其他节点进行恢复。

（3）领导者重选

Conterller Manager 和 Scheduler 在多个 Master 节点上运行，但不采用负载均衡模式。这些组件通过领导者选举机制，最终只有一个领导者处于活动状态，避免出现调度不一致问题。若当前领导者崩溃，则另一个组件会被选举为新的领导者，以此保持高可用性。

2.4 HAProxy+Keepalived 高可用技术特点

传统的高可用云平台通常采用 HAProxy 结合 Keepalived 的实现方案。这种架构模式虽然具有较长的应用时间，并支持多种负载均衡算法，但其配置复杂度较高，对服务器资源需求较大，且配置过程中容易出现错误，导致整体使用和维护成本相对较高。

（1）架构冗余：需独立部署负载均衡器与高可用代理，导致资源利用率低；

（2）动态适配不足：依赖脚本实现服务变更同步，延迟高且易出错；

（3）扩展性受限：BGP 等高级网络功能需额外配置，难以适配多云环境 [3]。

3 kube-vip 的工作原理。

通过安装 kube-vip 的服务器集群将会生成一个虚拟的 IP（VIP）地址，VIP 地址作为集群的统一入口。外部请求通过访问该 VIP 地址来与集群 API server 进行交互。kube-vip 利用分布式算法，在集群的各个节点之间进行状态同步和选举，确保在任何时刻都有一个主节点来负责处理与 VIP 相关的网络流量和管理任务。当主节点出现故障时，通过算法能快速选举出一个新的主节点，并将 VIP 的控制权转移到新主节点上，从而保证集群的服务连续性和可用性。同时，kube-vip 还会监控集群节点的状态，一旦发现节点异常或故障，会及

时进行相应的处理和调整，以维持整个 Kubernetes 集群的高可用性，使得应用能够稳定地运行在高可用的集群环境中，避免因节点故障而导致服务中断，原理如图 2 所示。

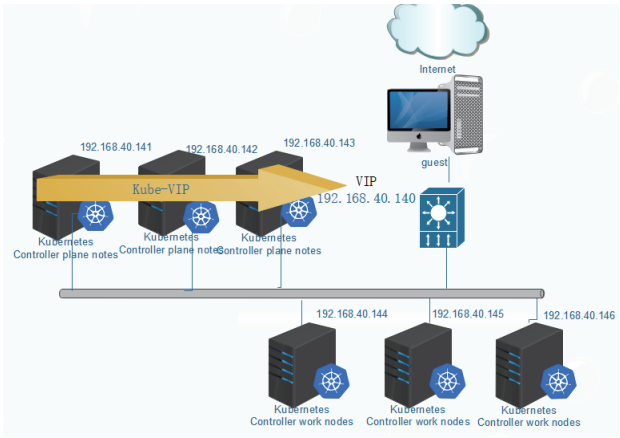

图 2 kube-vip 集群部署图

kube-vip 的理论创新体现在以下三个方面：

（1）控制平面深度集成：基于 Kubernetes Operator 模式，通过 Watch 机制实时感知 Service/Endpoint 变化，实现负载均衡规则动态更新；

（2）轻量级选举算法：采用 Raft 协议改进版，在保证一致性的前提下将选举耗时从秒级降至毫秒级；

（3）混合网络支持：通过 ARP（局域网）与 BGP（广域网）双模式自适应，支持跨数据中心流量调度。

综上所述，kube-vip 因其轻量级、无外部依赖和与 Kubernetes 高度集成的特点，成为实现 Kubernetes 高可用性和负载均衡的优先选择方案 [4]。

4 kube-vip 高可用平台设计与实现

本文使用 Kubeadm 部署 4 个节点的 Kubernetes 集群，其中 3 个 Master 节点及 1 个 worker 节点。kube-vip 运行在三个主节点的服务器上，减少了服务器的数量。kube-vip 集群将生成 192.168.40.140 的 VIP 地址，Kubernetes 工作节点服务器通过 VIP 地址加入集群中，并根据主节点的部署运行。Kubernetes 集群服务器配置如表 1 所示。

（1）按照集群服务器配置表要求准备好四台服务器，每台服务器都需要安装好 Ubuntu 操作系统，给 Ubuntu 操作系统配置好相应 IP 地址，关闭防火墙和 SELinux，确保全网互联互通 [5]。在所有服务器上添加容器软件 yum 源，安装 Containerd 容器，配置镜像加速，启动 Containerd 容器服务 [6]。配置流程如图 3 所示。

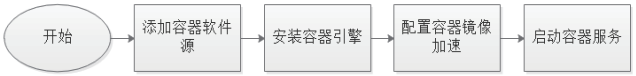

图 3 所有服务器安装 Containerd 容器环境

（2）在所有服务器上添加 Kubernetes 软件 yum 源，安装 Kubeadm、Kubelet 和 Kubectl 软件，并初始化主节点，如图 4 所示。

图 4 所有服务器上安装

（3）在第一台主节点上创建 kube-vip 配置文件，由于集群规模小，选择 ARP 部署模式，修改配置参数，并拷贝给另外两台主节点，如图 5 所示。

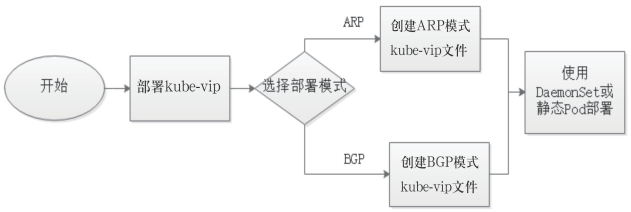

图 5 在第一个主节点服务器上创建 kube-vip 文件

（4）在第一台主节点上生成 kube-vip 静态 kube-vip.yaml 配置文件。

```
export VIP=192.168.40.140
export INTERFACE=ens33
ctr image pull docker.io/plndr/kube-vip:0.3.1
ctr run --rm --net-host docker.io/plndr/kube-vip:0.3.1 vip /kube-vip manifest pod --interface $INTERFACE --vip $VIP --controlplane --services --arp --leaderElection | tee  /etc/Kubernetes/manifests/kube-vip.yaml
```

（5）在第一台主节点上使用生成的 kubeadm.yaml 文件，进行 Kubernetes 初始化。

```
kubeadm init --upload-certs --config kubeadm.yaml
```

（6）在第一个主节点准备好后，加入其他主节点和工作节点。

表 1 Kubernetes 集群服务器配置表

节点名称	节点 IP	节点角色	CPU- 内存 - 磁盘	操作系统	VIP 地址
Master1	192.168.40.141	主节点 Master	2C/4 GB/100 GB	Ubutnu 22.04 LTS	192.168.40.140
Master2	192.168.40.142	主节点 Master			
Master3	192.168.40.143	主节点 Master			
Node1	192.168.40.144	工作节点 Work	2C/8 GB/200 GB		

其他主节点加入集群命令：

kubeadm join 192.168.40.140:6443 --token hash.hash --discovery-token-ca-cert-hash sha256:hash --control-plane --certificate-key key

工作节点加入集群命令：

kubeadm join 192.168.40.140:6443 --token hash.hash --discovery-token-ca-cert-hash sha256:hash

执行完成后集群就可以启动起来，在所有服务器上安装好网络插件，查看服务器节点状态为 Ready，如图 6 所示。

```
root@master1:~# kubectl get nodes
NAME      STATUS   ROLES           AGE   VERSION
master1   Ready    control-plane   21d   v1.28.1
master2   Ready    control-plane   21d   v1.28.1
master3   Ready    control-plane   21d   v1.28.1
node1     Ready    <none>          21d   v1.28.1
```

图 6 集群运行状态

5 kube-vip 高可用故障测试

按照 kube-vip 集群部署图，将对 Kubernetes 容器云平台高可用进行故障验证。

（1）集群正常运行状态

Kubernetes 集群正常运行时，在 Master1 节点上可以查看到高可用 VIP 地址，这时 ens33 有两个 IP 地址，其中 192.168.40.140 就是 VIP 地址，运行状态如图 7 所示。

```
root@master1:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:50:56:27:d3:2f brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.40.141/24 brd 192.168.40.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet 192.168.40.140/32 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe27:d32f/64 scope link
       valid_lft forever preferred_lft forever
3: kube-ipvs0: <BROADCAST,NOARP> mtu 1500 qdisc noop state DOWN group default
    link/ether ce:fb:02:ca:e7:23 brd ff:ff:ff:ff:ff:ff
    inet 10.101.150.103/32 scope global kube-ipvs0
       valid_lft forever preferred_lft forever
    inet 10.109.182.122/32 scope global kube-ipvs0
       valid_lft forever preferred_lft forever
    inet 10.96.0.1/32 scope global kube-ipvs0
       valid_lft forever preferred_lft forever
    inet 10.96.0.10/32 scope global kube-ipvs0
       valid_lft forever preferred_lft forever
```

图 7 在 Master1 节点上查看生成的 VIP 地址

（2）故障模拟测试

模仿故障，将第一个控制节点 Master1 进行关机，观察 VIP 地址会自动迁移到其他 Master 主机节点上，Kubernetes 服务不受影响。在 Master3 查看 IP 地址，这时 ens33 网卡上显示有两个 IP 地址，其中一个地址为节点原有的网卡地址，另一个为迁移来的 VIP 地址。切换时间小于 0.8 s，服务中断低于 1 s，运行状态如图 8 所示。

```
root@master3:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:50:56:3b:c2:95 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.40.143/24 brd 192.168.40.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet 192.168.40.140/32 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe3b:c295/64 scope link
       valid_lft forever preferred_lft forever
```

图 8 Master3 节点接管 VIP 地址

（3）集群状态查看

在 Master2 或者 Master3 节点查看节点状态，这时 Master1 状态为 NotReady，Master2 和 Master3 状态为 Ready，保证了 Kubernetes 集群正常运行。用户可以通过访问固定的 VIP 地址继续管理 Kubernetes 集群。运行状态如图 9 所示。

```
root@master2:~# kubectl get nodes
NAME      STATUS     ROLES           AGE   VERSION
master1   NotReady   control-plane   21d   v1.28.1
master2   Ready      control-plane   21d   v1.28.1
master3   Ready      control-plane   21d   v1.28.1
node1     Ready      <none>          21d   v1.28.1
```

图 9 Kubernetes 集群正常使用

6 结语

针对目前配置 Kubernetes 高可用集群时出现的部署流程繁琐和配置难度大，资源开销高等突出问题，提出了基于 kube-vip 原生的 Kubernetes 高可用集群解决方案，并进行了工程实践。实践结果表明，在 Kubernetes 运行环境中，只需通过一个 kube-vip 配置文件就能实现集群高可用，简化了配置流程。该方案成本低、易维护，显著提升了平台可靠性，有力支撑了容器云平台运行。未来将应用到学院教学实训平台中，提供稳定、可靠的基础设施支撑，显著提升业务连续性和抗风险能力，同时让学生学习和实践企业级的云平台运维与高可用架构，培养核心技能。

参考文献：

[1] 武志学 . 云计算虚拟化技术的发展与趋势 [J]. 计算机应用 , 2017, 37(4):915-923.

[2] 张杨 . 基于 Kubernetes 的容器云平台的高可用研究与实现 [D]. 成都 : 电子科技大学 ,2023.

[3] 许彪 , 王湘渝 , 朱爱梅 . 基于 Mariadb Galera 的高可用数据库集群技术 [J]. 信息技术与信息化 ,2021(10):25-27.

[4] 郭雷 , 毛玲燕 . 基于 Kubernetes 的企业级容器云平台设计与实践 [J]. 信息技术与标准化 ,2022(9):73-77.

[5] 王湘渝 , 邱春荣 .Docker 技术在构建 Linux 实验平台中的应用 [J]. 教育现代化 ,2019,6(46):120-123.

[6] 代强 , 赵亦辉 . 基于容器的科学工作流云计算平台 [J]. 计算机时代 ,2023(12):5-8.

【作者简介】

王湘渝（1974—），男，湖南长沙人，硕士，教授，研究方向：云计算、计算机网络和职业教育。

（收稿日期：2025-09-05 修回日期：2026-04-08）