IAS2023

Pingmesh:

在基础网络和容器网络中的可观测性实践

董江 / 中国移动云能力中心 高级系统架构专家

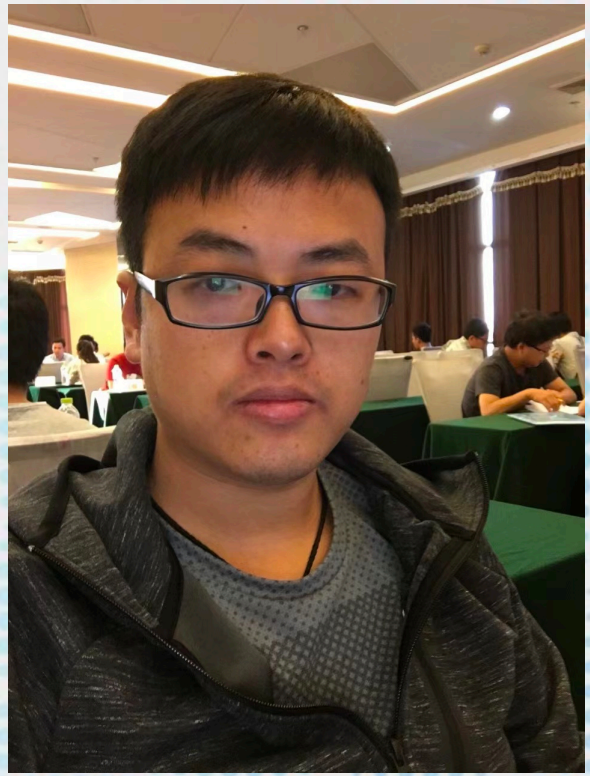
个人介绍

董江 中国移动云能力中心 高级系统架构专家

容器技术布道者及实践者，ServiceComb微服务框架核心开发者， 云原生社区Member、Prometheus社区 PMC、Alibaba社区Collaborator、华为云MVP/HCDE

KubeService Stack社区发起 https://stack.kubeservice.cn/
个人博客 https://kubeservice.cn/

曾就职于 百度、阿里、滴滴、华为、贝壳



01

场景

基础设施复杂

基础设施 – 容器基础设施“多”而“杂”

集群数 XXX个

Node数 XXXXX个

实例数 XX万个

对外产品
- 容器服务KCS
- RDS
- 对象存储EOS
- SaaS类

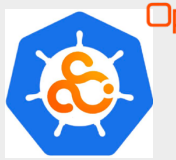
对内产品
- 内部IT平台
- Console
- 边缘管理
- 数据服务

容器
- 数据库
- 缓存
- 中间件
- 网关
- 容器引擎

IaaS
- 数据中心（机房、上电）
- 基础硬件（PM、VM、Storage）
- 网络平面（中心、区域）

- 运维平台
- 可观测性平台
- 成本核算
- 稳定性

基础设施 – 国产化适配国产化 软硬件 和 OS系统



容器引擎

Operating System

Instruction Set

GPU

对操作系统OS适配：

- openEuler
- Anolis OS
- BCLinux

对GPU适配：

- NVIDIA T4/A100/H100
- HUAWEI昇腾910B

对CPU指令集适配：

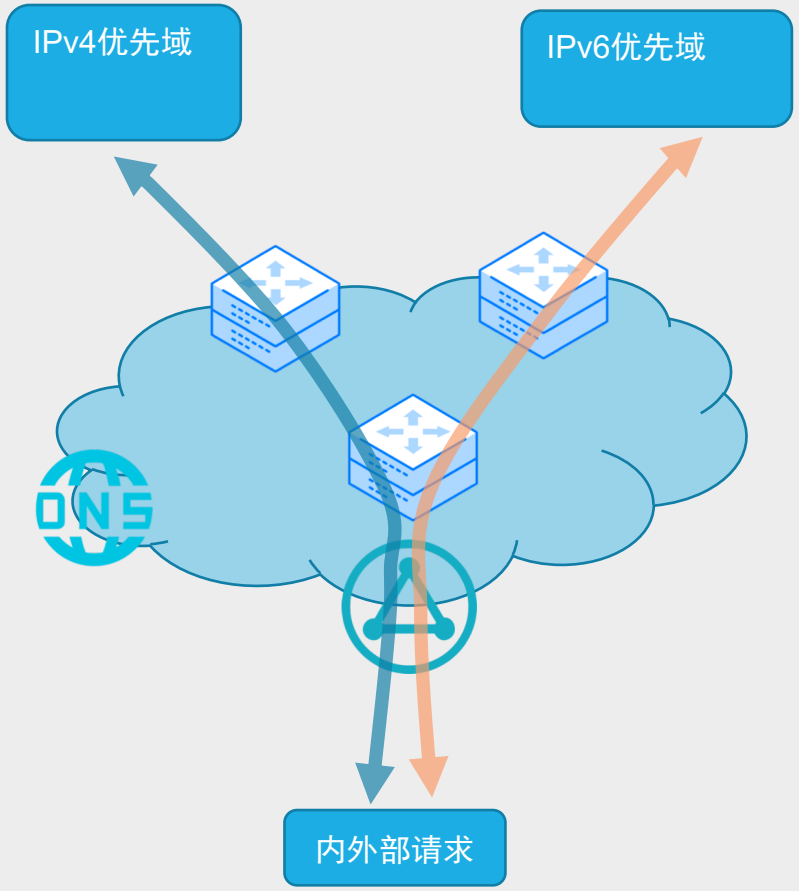
- amd64/x86_64
- arm64/aarch64
- SW
- LoongArch

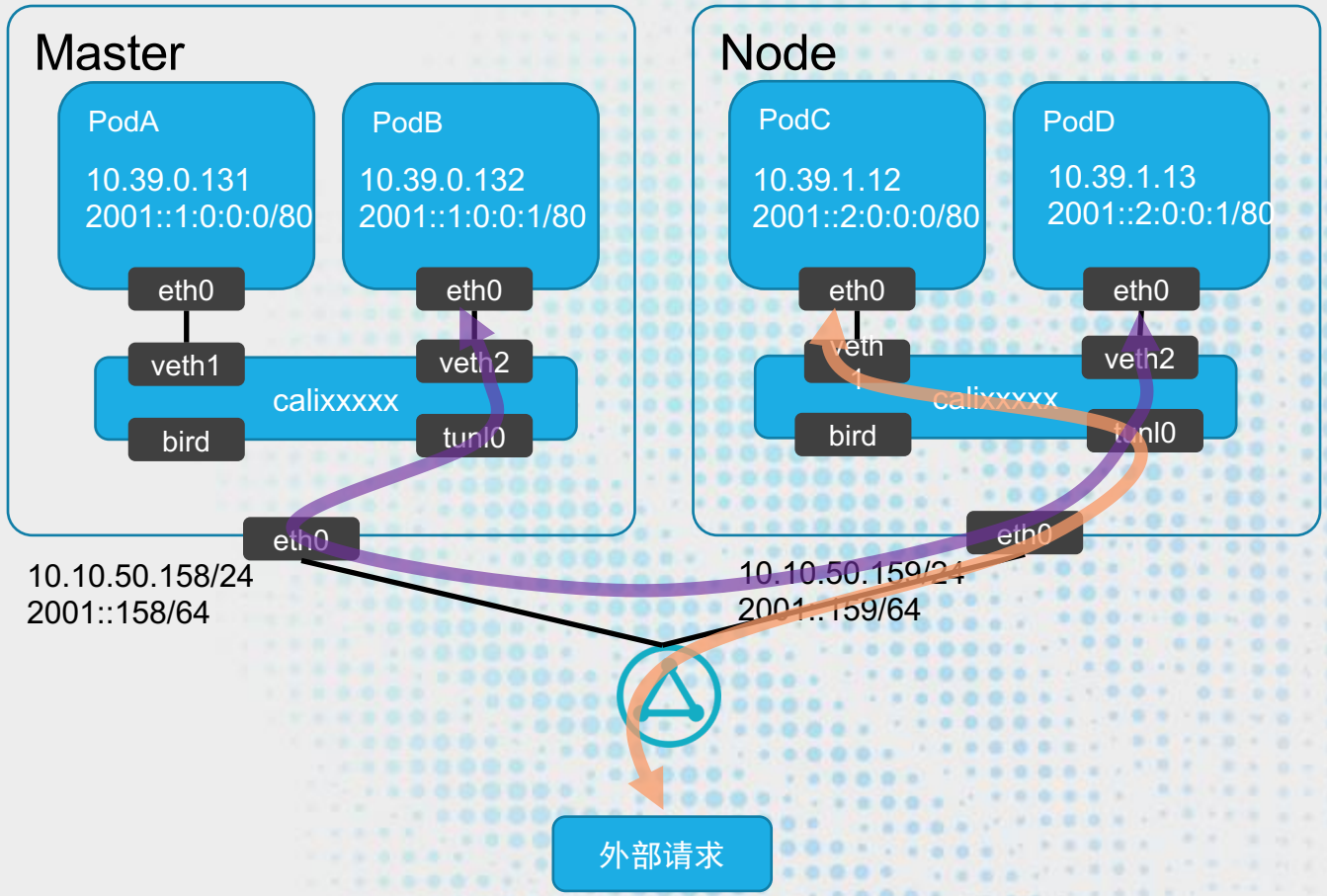
生态适配国产化：

- Runtime：kata、docker和containerd
- CSI：ceph-csi、nfs-csi、ebs-csi、Heketi csi、pmem csi
- CNI：calico、cilium
- Addon：Prometheus、Grafana、OpenTracing

基础设施 – 适配IPv4/IPv6双栈

IPv6改造： 按照IPv4单栈 -> IPv4/IPv6双栈 -> IPv6单栈 路线进行平滑过渡。



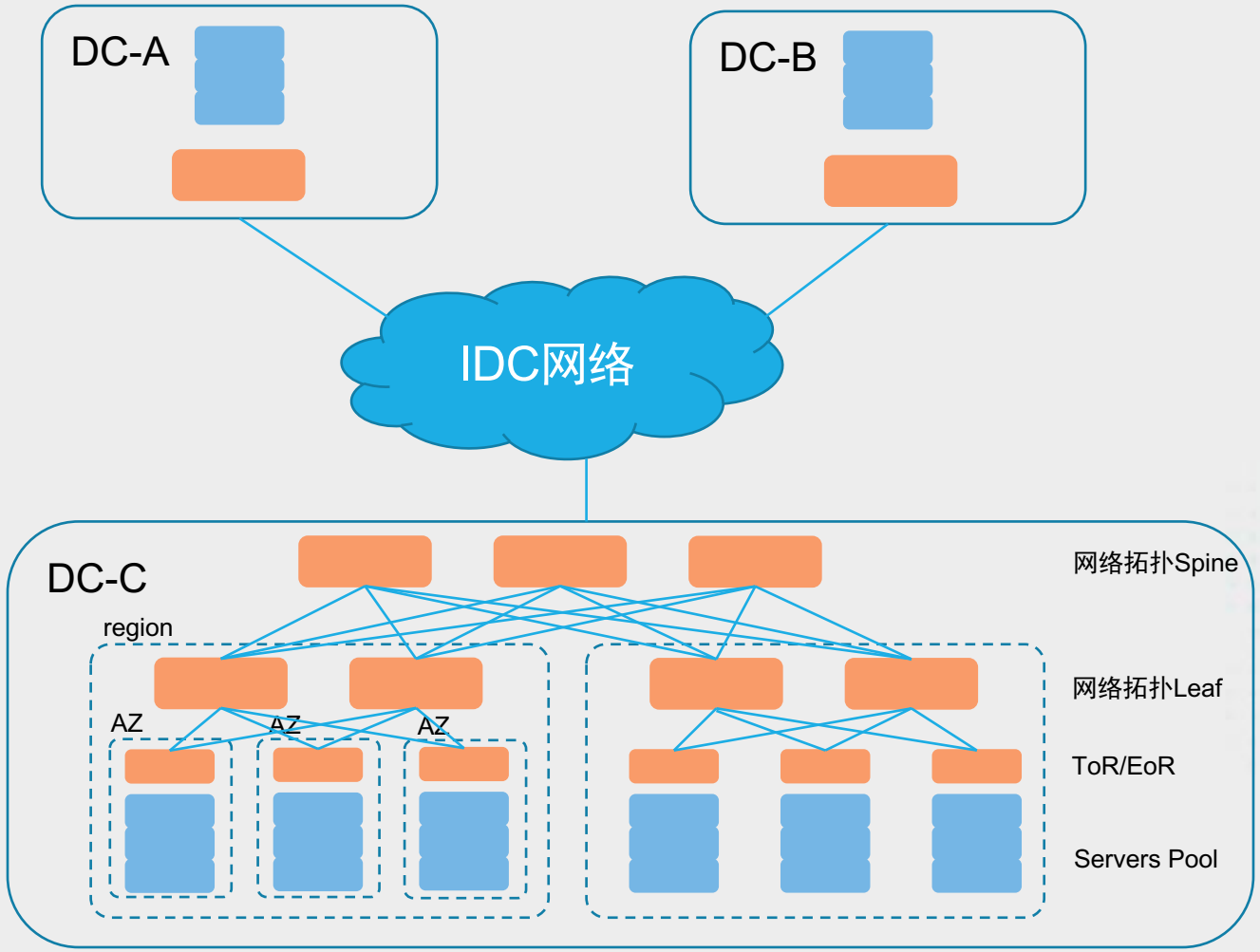


02

Pingmesh对基础网络和容器网络检测

- 网络环境
- 业绩优秀案例
- 实现架构
- 核心技术点

网络环境 – 4 + N + 31 数据中心互联互通

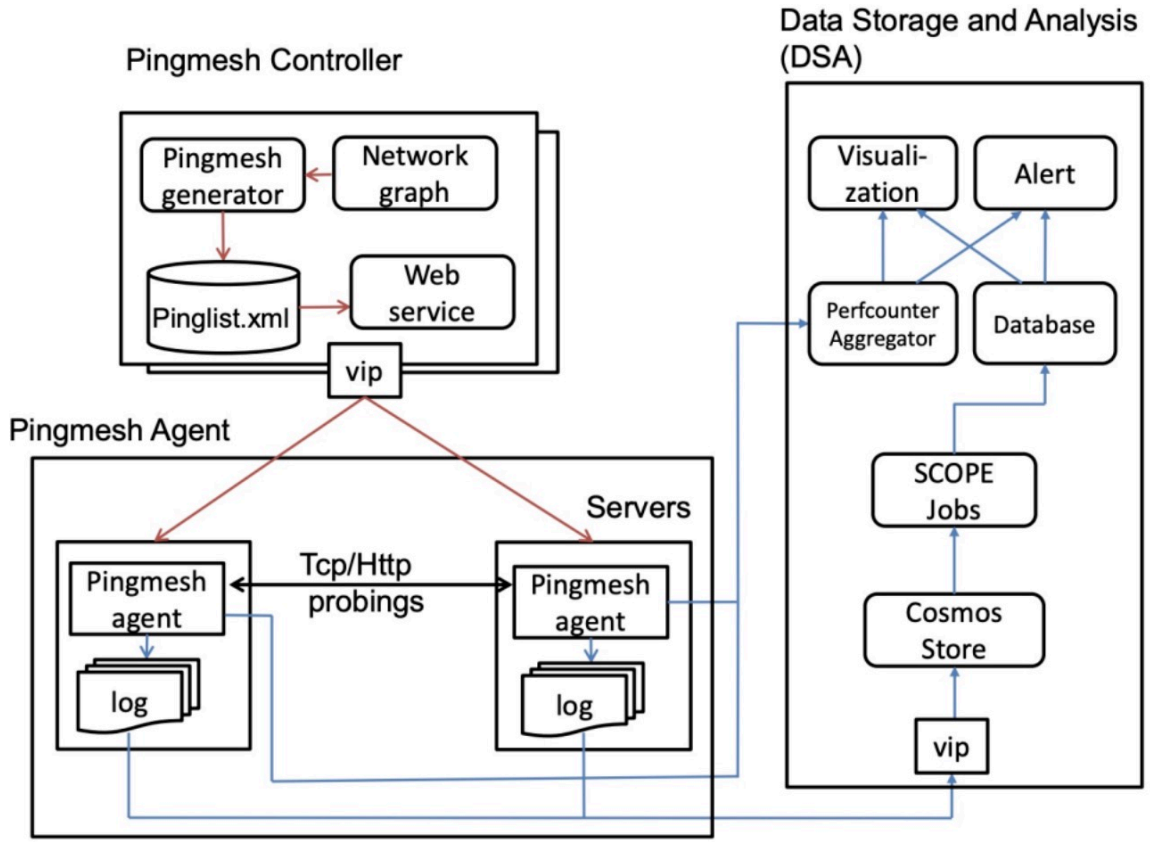


【4 + N + 31】数据中心
- 4 热点区域
- N 中心节点
- 31 省级节点

【网络拓扑】
- 叶脊网络(Spine-Leaf)
- 成百上千的节点、网卡、交换机、路由器
- 无数的网线、光纤

【引出】：
a. 如何判断一个故障是网络故障？
b. 如何定义和追踪网络的 SLA？出了故障如何去排查？

业绩优秀案例 - 微软 Pingmesh 论文借鉴



举个例子：

IDC设备： 10000台

Ping Task： 10000 * (10000-1) = 10000*10000

如果是每30s进行一次ping， 一次ping 需要 payload大小是64bytes 数据存储

每天存储数据量：

10000*10000*24*60*2*64 = 1.8432e+13 bytes

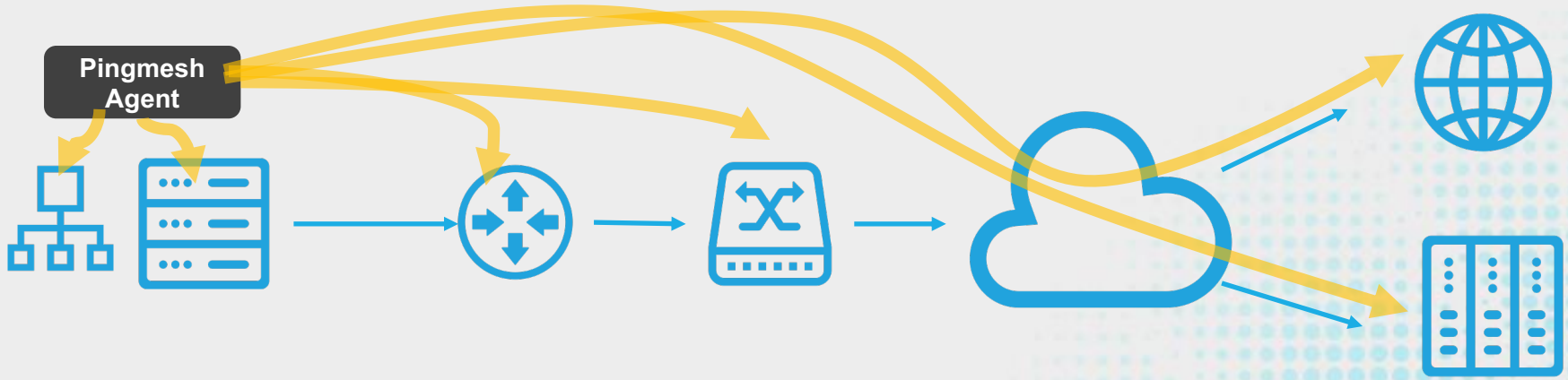
任务数 次数 数据大小

= 16.7638 TB

? 如果只记录fail和timeout的记录，可以节约99.999%的存储空间

参考：《Pingmesh: A Large-Scale System for Data Center Network Latency Measurement and Analysis》

实现思路 – 结合用户场景和需求，设定大方向



【两个问题】
- 如何判断一个故障是网络故障？
- ~~如何定义和追踪网络的 SLA？~~出了故障如何去排查？

【客户】
- 成本敏感

【场景】
- 容器网络 & 主机网络
- 公有云 & 私有云场景
- 国产化场景

【能力集】
- 用户场景
- 全局性

【Logging vs Metrics】：
大数据实时流 vs 监控指标

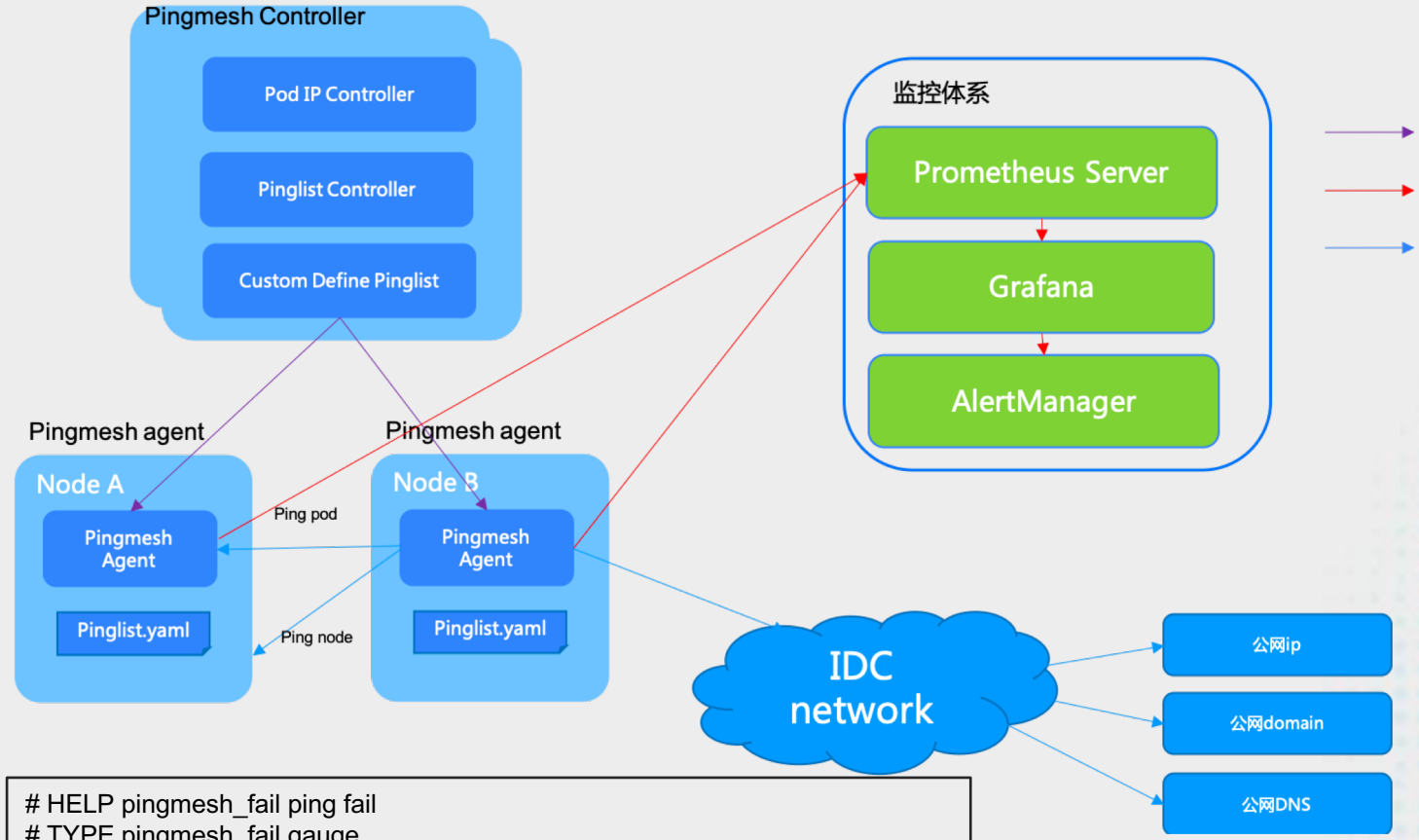
【Ping 模式支持更广泛模式】：
DNS 、ICMP 、SCMP、IP、Domain、SNMP等

【Ping场景聚焦】：
容器、主机网络场景；
多网卡IP、IPv6/v4双栈场景

【手动探测】：
DryRun 和 手动网络探测验证

【资源】
高性能、低能耗、过载保护、全局轮训

架构实现 – 容器网络监控Pingmesh

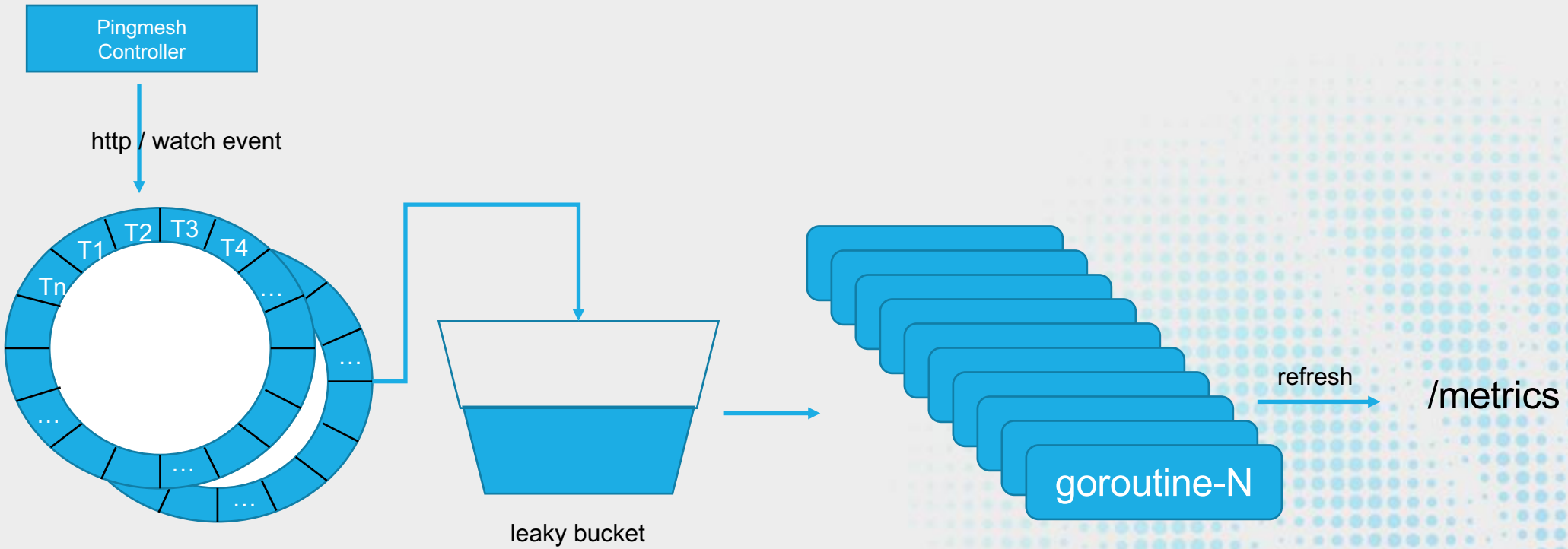



```
# HELP pingmesh_fail ping fail
# TYPE pingmesh_fail gauge
pingmesh_fail{target="8.8.8.8",tor="ping-public-demo"} 1

# HELP pingmesh_duration_milliseconds duration of ping rtt
# TYPE pingmesh_duration_milliseconds gauge
pingmesh_duration_milliseconds{target="docker.io",tor="ping-public-demo"} 245

# HELP pingmesh_warning alertmanager
# TYPE pingmesh_warning gauge
pingmesh_warning 1
```

```
setting:
  # the maximum amount of concurrent to ping, uint
  concurrent_limit: 20
  # interval to exec ping in seconds, float
  interval: 60.0
  # The maximum delay time to ping in milliseconds, float
  delay: 200
  # ping timeout in seconds, float
  timeout: 2.0
  # send ip addr
  source_ip_addr: 0.0.0.0
  # send ip protocal
  ip_protocol: ip6

mesh:
  - name: ping-public-demo
    type: OtherIP
    tasks :
      - 114.114.114.114
      - 8.8.8.8
      - 240C::6666
      - operatorhub.io
      - docker.io
      - kubernetes.default.svc.cluster.local
```

技术实现：基于blackbox构建的Pingmesh体系

核心技术点： Agent资源限制 和 性能限制



基于Prometheus社区blackbox-exporter实现：

- 双buffer存储task任务，确保一轮任务完全完毕；
- 漏桶确保task任务，在一定时间窗口window下，均衡执行完成；
- 限制Agent全局协程总数，控制内存和CPU使用；
- 内存refresh周期，确保数据周期更新；

容器网络下： 10000+ 任务，单个agent资源消耗

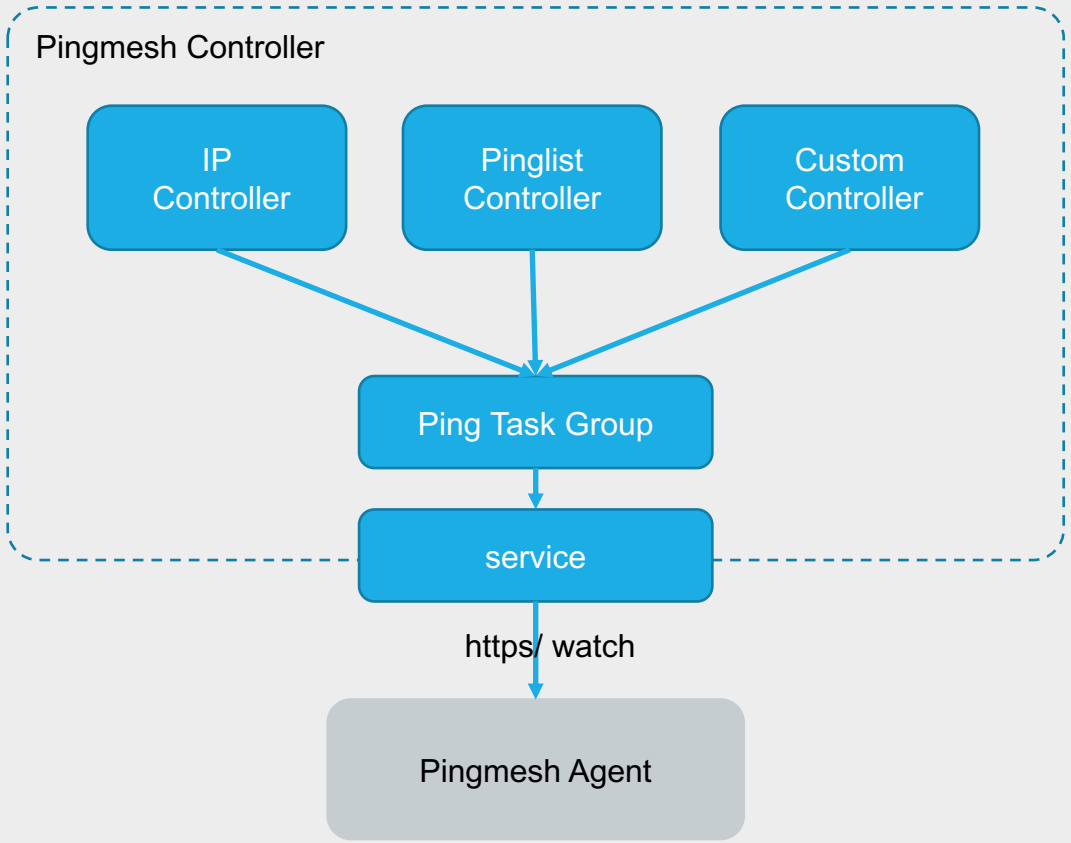
```
metrics-prometheus-pingmesh-exporter-4sl1n   3m   26Mi
metrics-prometheus-pingmesh-exporter-7r8t8   3m   45Mi
metrics-prometheus-pingmesh-exporter-9w44n   9m   27Mi
metrics-prometheus-pingmesh-exporter-kj7w9   6m   39Mi
metrics-prometheus-pingmesh-exporter-qplk4   4m   24Mi
metrics-prometheus-pingmesh-exporter-szlbx   2m   29Mi
```

10000 Node(32C64G)集群中， 资源总共消耗：

- CPU: 10000 * 10m = 100C， 占比 0.015%
- 内存: 10000 * 40Mi = 400Gi , 占比 0.0625%

技术实现：基于blackbox构建的Pingmesh体系

核心技术点： Pingmesh Controller 规划任务矩阵



【IP Controller】：
通过IP Controller自动获取到整个集群的podIP 和 nodeIp list
Pinglist controller：
通过Pinglist Controller 配置外部随机选择算法； 可以自定义二开；

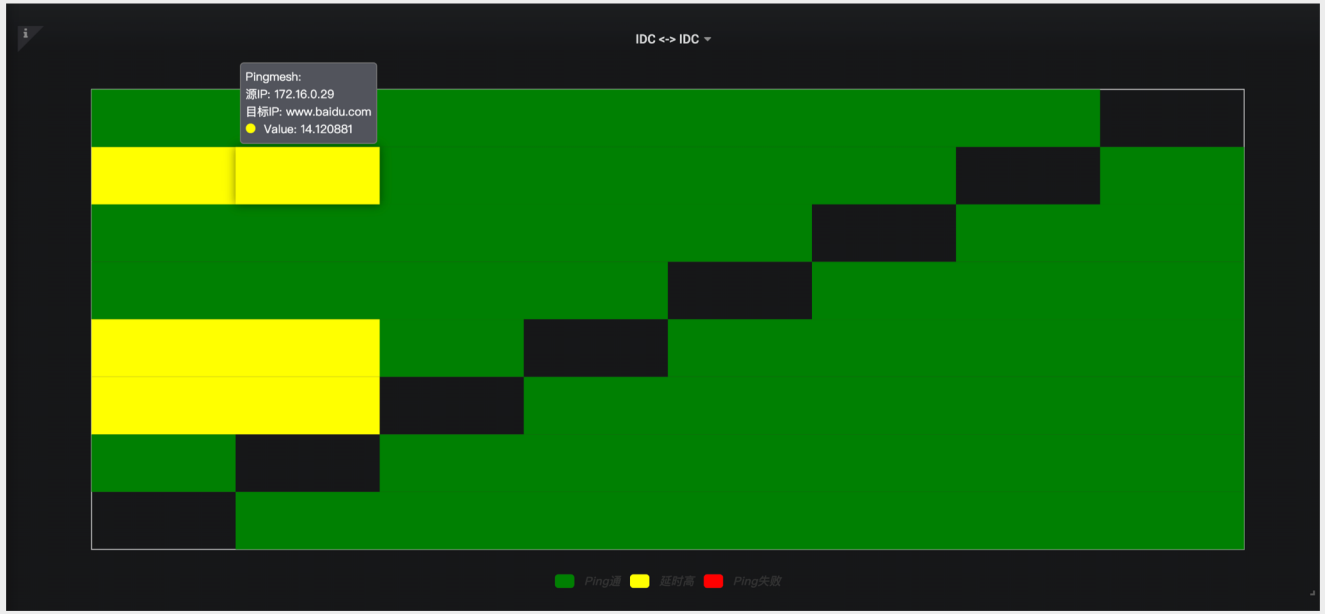
【Custom Controller】：
通过Custom Define Pinglist 在 pinglist.yaml 文件中补充 外部地址。
支持dns地址、外部http地址、domain地址、ntp地址、Kubenetes apiserver地址等等

pingmesh.yaml

```yaml
modules:
  dns:
    dns:
      preferred_ip_protocol: ip4
      query_name: kubernetes.defalut.svc.cluster.local
      transport_protocol: tcp
    prober: dns
  dns_ipv6:
    dns:
      preferred_ip_protocol: ip6
      query_name: kubernetes.defalut.svc.cluster.local
      transport_protocol: tcp
    prober: dns
  http_2xx:
```

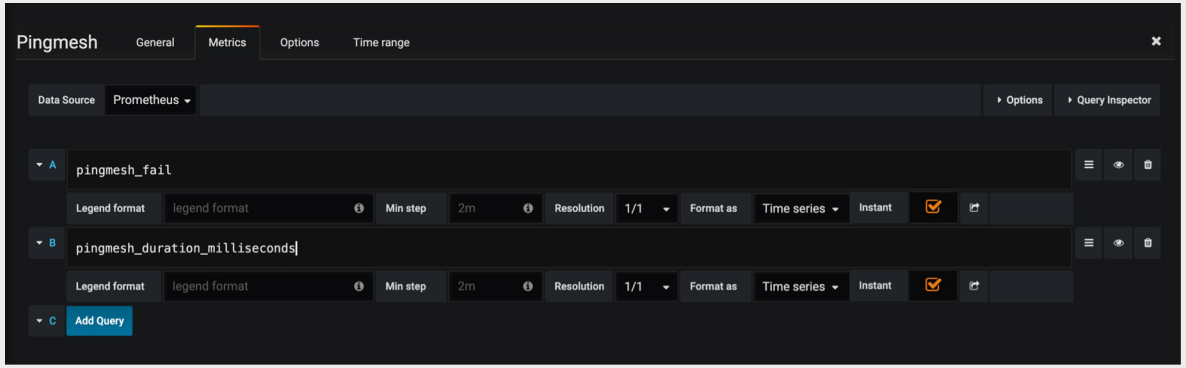
【PingTaskGroup】：
规划任务Group集合和权重； watch按group下发事件；

核心技术点： Pingmesh Heatmap Panel



满足Pingmesh Heatmap panel实现：

- 可配置 pingmesh heamap panel 有向图
- 可设置呈现timeout、fail 任务呈现颜色
- 支持自定义数据图表维度



技术实现：Grafana Panel Plugin: Pingmesh Heatmap Panel

03

未来展望

展望

随着AIGC的蓬勃发展，DPU和RDMA等技术已经获得广泛应用。 对以下要求更高。

- 方式灵活
- 快速精准
- 全面覆盖
- 成本优化

欢迎试用 & 共建：

https://artifacthub.io/packages/helm/prometheus-community/prometheus-pingmesh-exporter
https://github.com/prometheus-community/helm-charts/tree/main/charts/prometheus-pingmesh-exporter

https://github.com/kubeservice-stack/pingmesh-agent

https://grafana.com/grafana/dashboards/?search=pingmesh
https://github.com/kubeservice-stack/pingmesh-heatmap-panel

Thanks