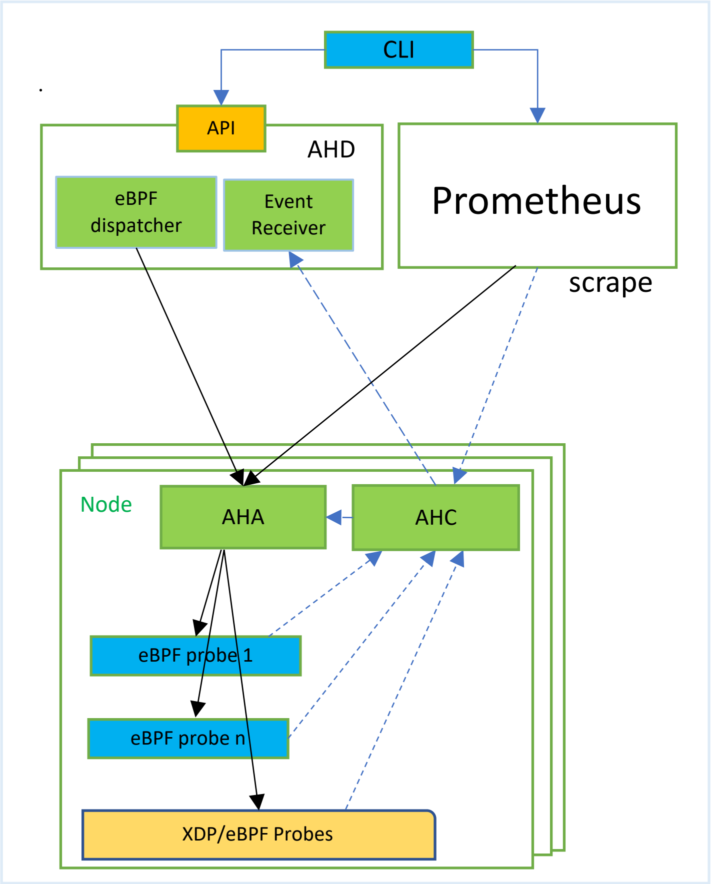
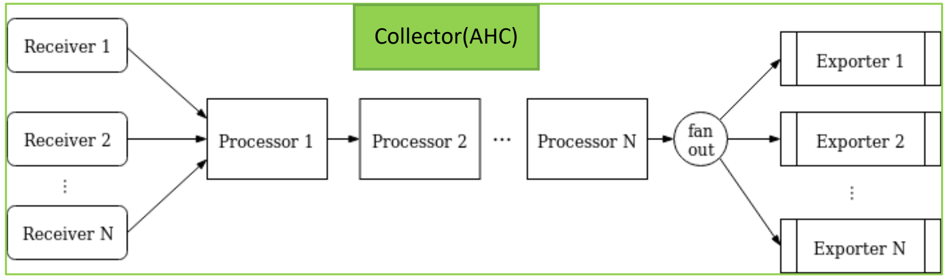
Aeolus: eBPF based monitoring framework for Container network

Wei Yue
Cloud Lab
Futurewei Technologies

V0.1
12/2022

Aeolus work with K8s and Prometheus





AHD: Aeolus Health Dictator;
AHA: Aeolus Health Agent;
AHC: Aeolus Health Collector

Aeolus includes three major components:

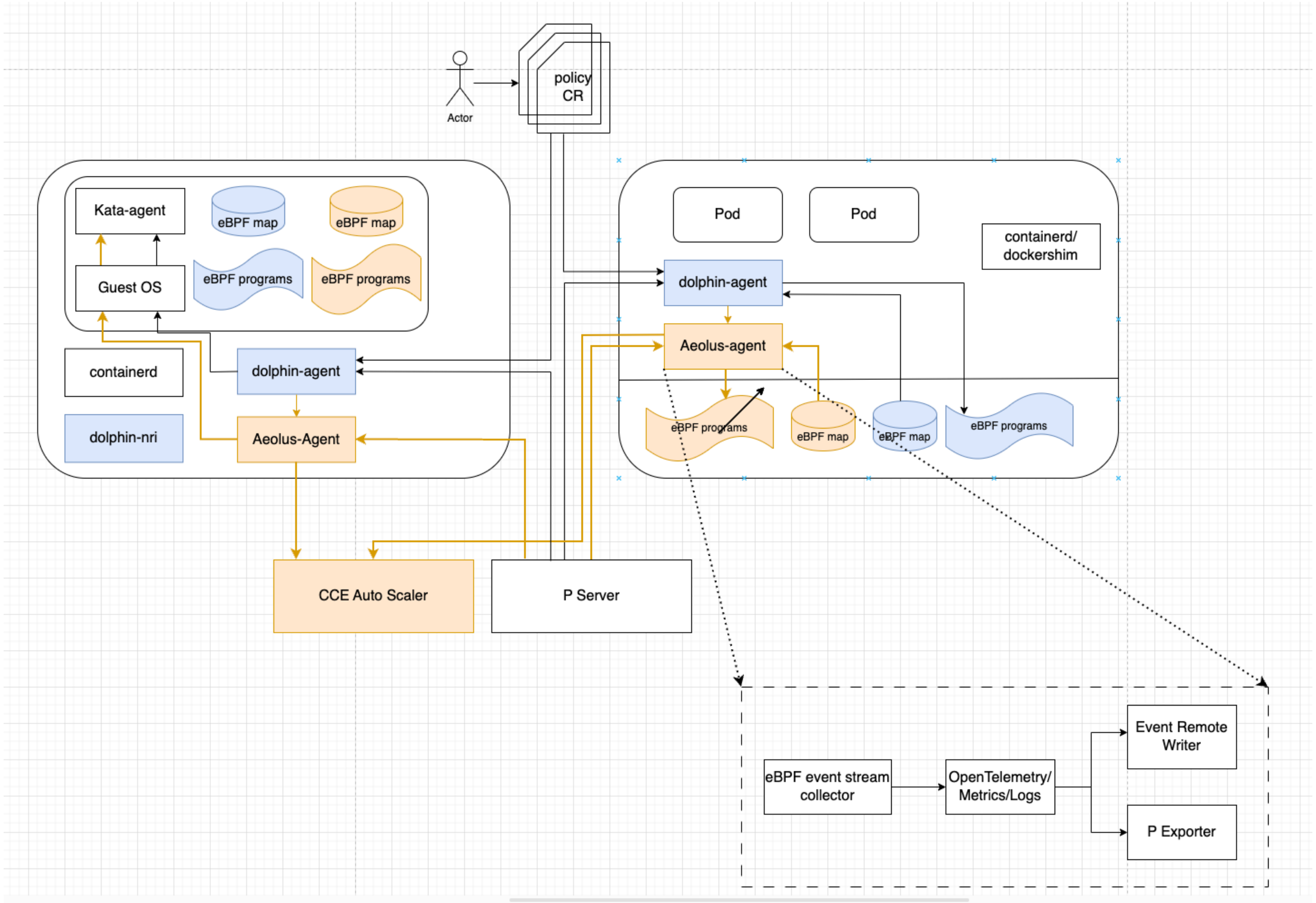
- **AHC** is literally an *open telemetry collector*. The collector can be configured to have one or more pipelines. Each pipeline includes:
    - a set of Receivers that receive the data
    - a series of optional Processors that get the data from receivers and process it
    - a set of Exporters which get the data from processors and send it further outside the Collector.

By default, one exporter(e.g. P Remote Writer Exporter) is needed to push event data to AHD's event receiver; another exporter can be configured for Prometheus;

- **AHA** is deployed on needed node as daemon-set, it receives instruction from AHD and/or optionally from Prometheus and injects eBPF snippet to specific hooks;

- **AHD** is responsible for receiving triggered event data from AHC and provide mechanism to dynamically generate eBPF snippet based on the response and deploy to AHA. It is a critical and intelligent part for push based live anomaly detection and reaction which is not possible in pull-based Prometheus framework.

wyue@futurewei.com



Actor
policy CR
Kata-agent
eBPF map
eBPF map
Guest OS
eBPF programs
eBPF programs
containerd
dolphin-agent
dolphin-nri
Aeolus-Agent
Pod
Pod
containerd/ dockershim
dolphin-agent
Aeolus-agent
eBPF programs
eBPF map
eBPF map
eBPF programs
CCE Auto Scaler
P Server
eBPF event stream collector
OpenTelemetry/ Metrics/Logs
Event Remote Writer
P Exporter

合作点1：针对云原生场景基于eBPF的高定制化高性能优化方案

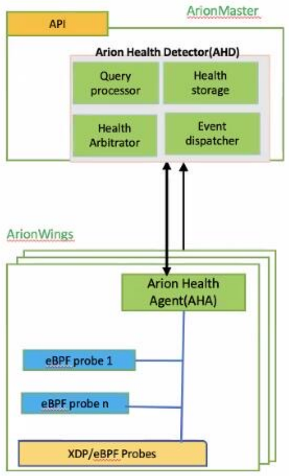
• 痛点：

1）serverless场景下每个Pod内都有个sidecar（如Knative中的Queue-proxy），cpu占用率高，与租户业务竞争资源。

2）当前云原生serverless监控基于轮询上报（容器团队dolphin，cilium，pixie），开销较大，延迟在秒级至分钟级，无法第一时间上报并响应，而当前serverless应用逐渐对毫秒级的性能异常敏感；

3）当前监控框架（如容器团队dolphin，cilium开源版）大多无法做到运行时修改，不够灵活，pixie可以动态部署但以数值监控为主，对于事件监控定制化支持度不高；（举个例子）

4）目前对于eBPF map的访问没有限制，任何其他应用都可以修改，存在安全风险。(验收方法)



基于eBPF高可编程性设计的高定制化高性能的event监控平台

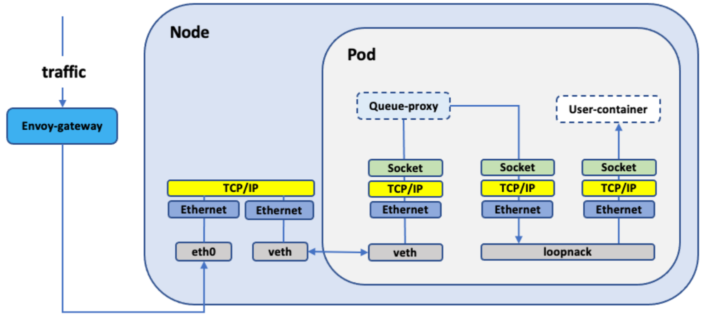
• 业务价值：

1）更低的响应延迟： sidecar延迟大幅降低，监控任务执行达到毫秒级响应；

2）更低的CPU开销：通过eBPF探针执行sidecar优化和监控，CPU资源消耗大幅降低；

3）更灵活的事件监控：定制化监控event，并支持动态创建与部署监控event；

4）保障eBPF资源的安全访问。



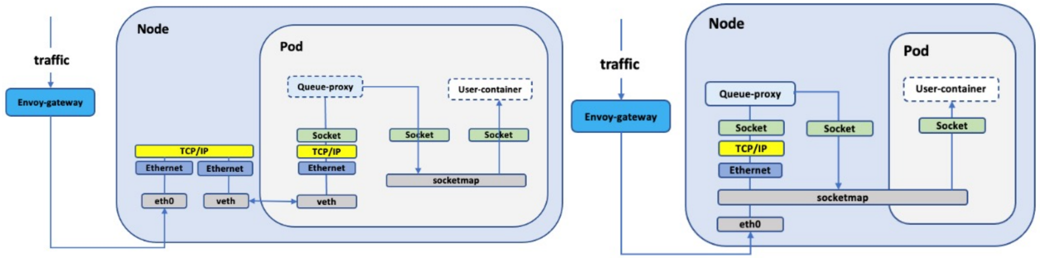
当前设计 – sidecar QP without eBPF acceleration

• 目标：

1）进一步探索利用eBPF等技术优化serverless的sidecar(如Knative中的Queue-proxy组件),搭建快慢路径, 延迟降低25%,CPU利用率比降低25%以上；

2）搭建基于eBPF探针的异常检测平台,延迟降低到10-100ms量级,CPU利用率比传统轮询方案降低50%以上；动态部署更加灵活性及定制化

3）**容器内eBPF map被异常修改可以被制止。**



两种基于eBPF的serverless组件优化方案探索

• 技术挑战：

1）优化sidecar性能和资源消耗的同时保证安全性不劣化；

2）针对不同监控场景，需找到最优的内核系统调用作为监控挂载点，以最小化监控开销；

3）内核态eBPF探针发现异常后，需快速和用户态agent交互并上报控制器。