Contribution ID: **85** Type: **not specified**

Traffic footprint characterization of workloads using BPF

Wednesday 11 September 2019 10:45 (45 minutes)

Application workloads are becoming increasingly diverse in terms of their network resource requirements and performance characteristics. As opposed to long running monoliths deployed in virtual machines, containerized workloads can be as short lived as few seconds. Today, container orchestrators that schedule these workloads primarily consider their CPU and memory resource requirements since they can easily be quantified. However, network resources characterization isn't as straight forward. Ineffective scheduling of containerized workloads, which could be throughput intensive or latency sensitive, can lead to adverse network performance. Hence, I propose characterizing and learning network footprints of applications running in a cluster, which can be used while scheduling them in containers/VMs such that their network performance can be improved.

There is a well-known network issue, which is achieving low latency for mice flows (those that send relatively small amounts of data) by separating them from the elephant flows (those that send a lot of data). I've written an eBPF program in C that runs at various hook points in the Linux connection tracking (aka conntrack) kernel functions in order to detect network elephant flow, and attribute them to the container or VM, where the flows ingress or egress from. The agent that loads this eBPF program from user space runs in every host in a cluster. It then feeds this learnt information to a container (or VM) scheduling system such that they can use this information proactively, while scheduling containerized workloads with light network footprint (e.g., microservices, functions) and heavy network footprint (e.g., data analytics, data computational applications) on the same cluster, in order to improve their latency and throughput, respectively.

eBPF facilitates running the programs with minimal CPU overhead, in a pluggable, tunable and safe manner, and without having to change any kernel code. It's also worthwhile to discuss how the workload's learnt network footprint can be used for dynamically allocating or tuning Linux network resources like bandwidth, vcpu/vhost-net allocation, receive-side scaling (RSS) queue mappings, etc.
I'll submit a paper with the (working) source code snippets and details if the talk is accepted.

I agree to abide by the anti-harassment policy

Yes

I confirm that I am already registered for LPC 2019

Primary author: GHAG, Aditi (VMware)

Presenter: GHAG, Aditi (VMware)

Session Classification: Networking Summit Track