LINUX PLUMBERS CONFERENCE September 20-24, 2021

Container tracing

Yordan Karadzhov

VMware Inc. - OSTC

How do we define a container?

a. This thing created by Docker
b. Something that is compliant withe the OCI Runtime Specification
c. A bunch of processes isolated in namespaces and governed by cgroups

Tracing containerized workloads sounds attractive, but we first have to agree what we mean by this.

Hooking to a container that is being created

- We need the PID of the parent process of the container
- And we need it as early as possible.

System calls involved in setuping the container:

1. **mkdir()** : to create the cgroups. Can filter for new directories in /sys/fs/cgroup/*
2. **unshare()** : to move the parent process into new namespaces. *
   - Alternatives: **clone()** and **setns()**
3. **pivot_root()** : to changes the root mount. *

* The caller is the parent process of the container.

Hooking to a running container:

1. Examine the cgroups. For example, look in
   /sys/fs/cgroup/<some cgroup>/docker/<container id>/tasks
2. Examine the Docker runtime
   /run/containerd/io.containerd.runtime.v2.task/moby/<container id>/init.pid
   and find all child processes.
3. Retrieve the layers of the container image from
   /proc/<container parent pid>/mounts
   If we **brute-force** /proc we can get the list of running container images.
   Some extra work is needed to separate the different instances of the same container.

- Anything else we can do?
- Is there a standard we can use?

Some ideas on what we want to trace:

- Files a container opens
  - Files a container reads
  - Files a container writes to
- What programs are executed
- What libraries are utilized
- Networks a container connects to
- How to follow a service through a container?
- How do containers on different nodes interact?
- **Anything else?**