Extend Cgroup Stats Collection and Interface with BPF

Hao Luo

Background

We collect kernel performance stats for jobs, which are running in containers, and export the stats to userspace via cgroupfs.

Stats definitions were

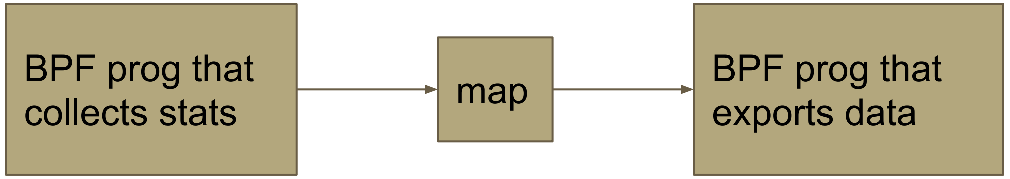
- implemented in kernel;
- tightly integrated with cgroup subsystem.

Motivation

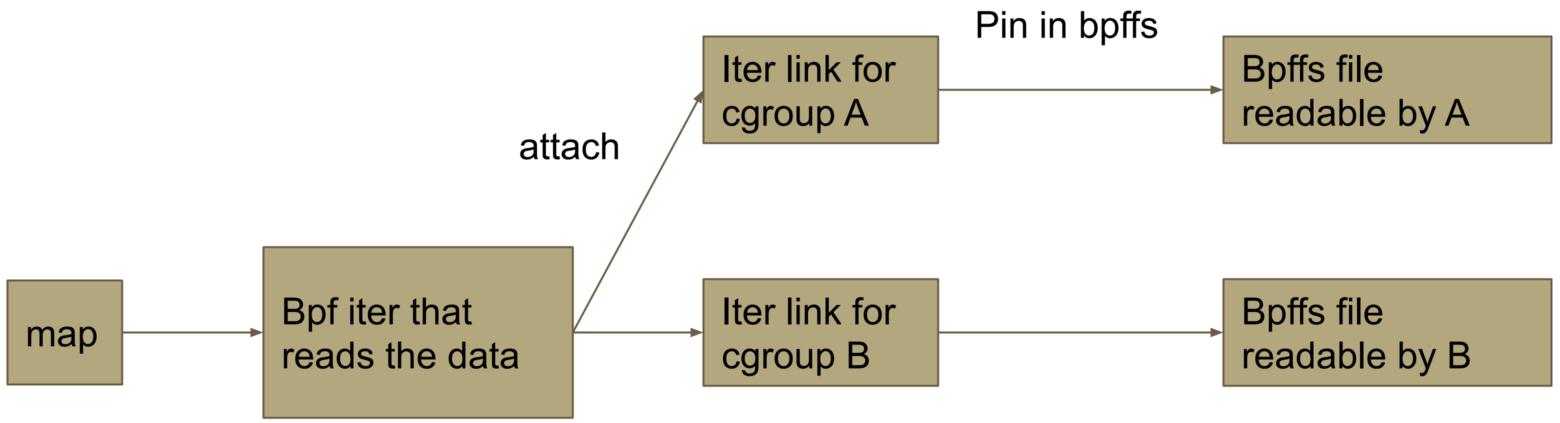
We want to migrate our internal per-cgroup stats to BPF.

High Level Design

A pipeline consisting of two components: collection and exportation



High Level Design - Export



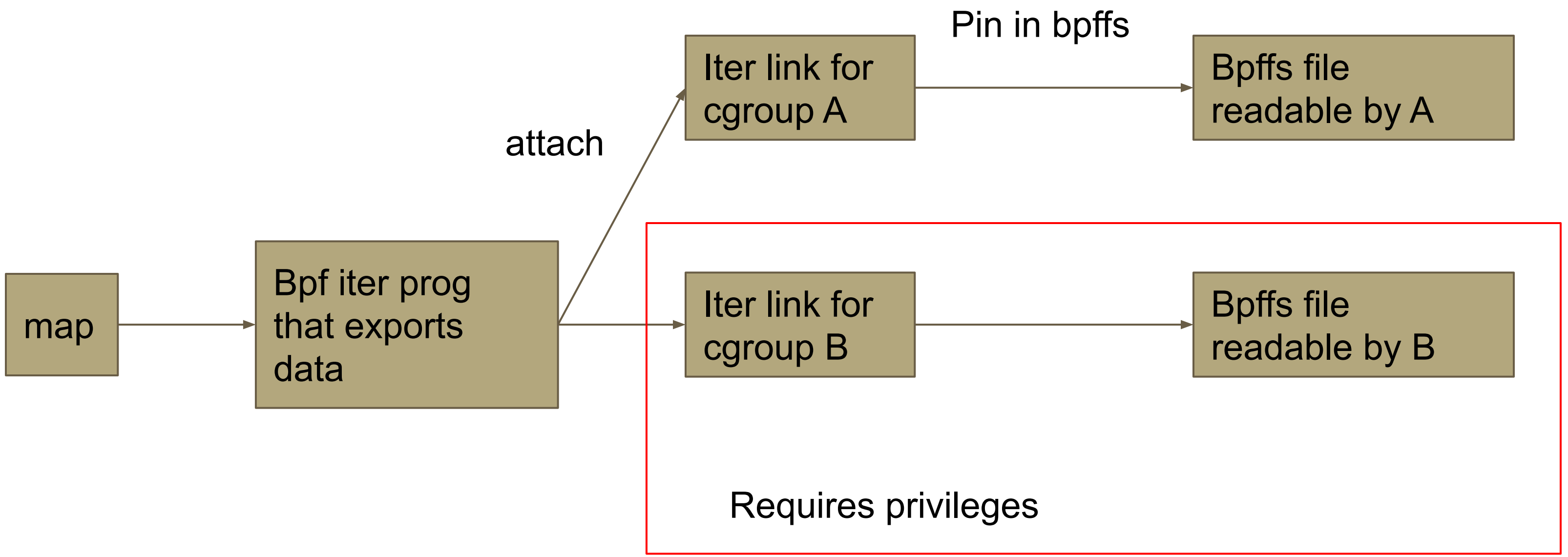
What’s the problem?

Unpriv users can create cgroups, but they can’t create iter_links and pin the links in bpffs (BPF disabled for unpriv users)

=>

When unpriv users creates cgroups, they can’t create BPF-based stats file without help from priv processes.

High Level Design - Export

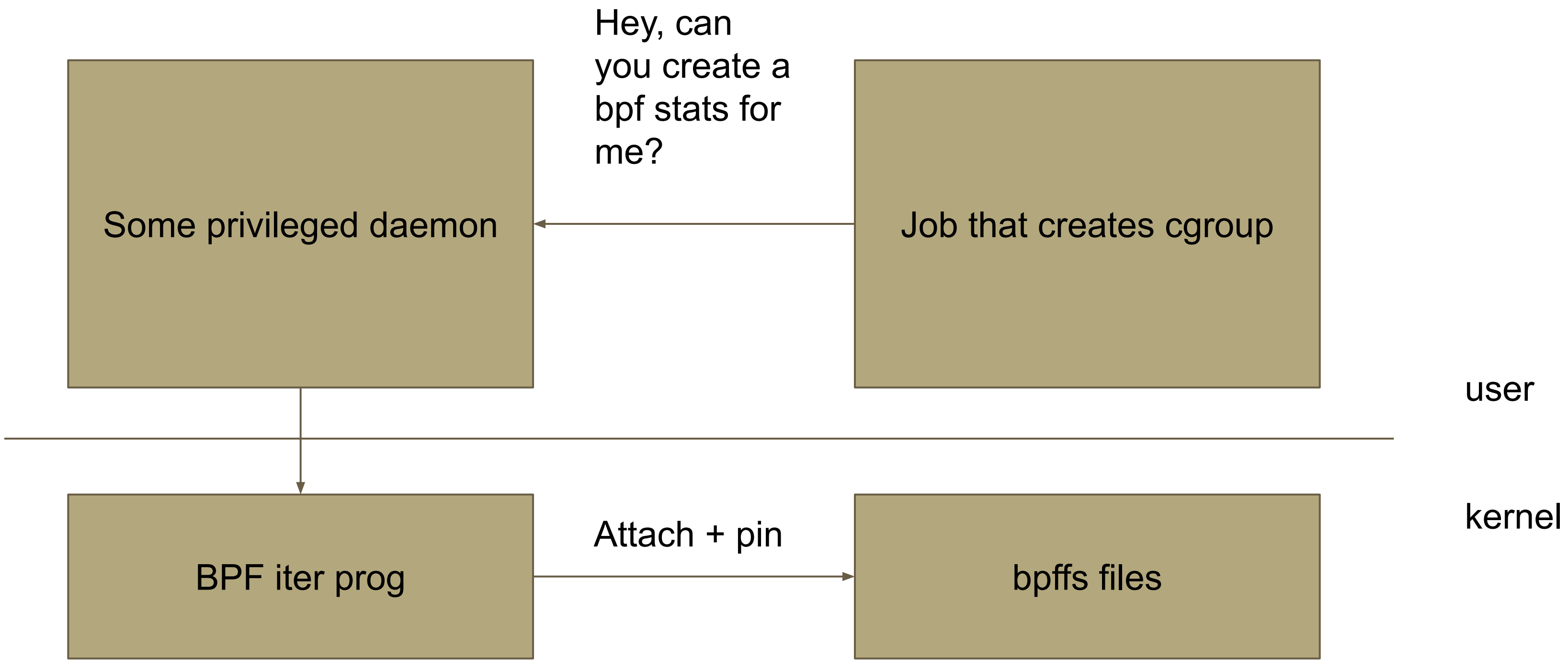


Idea I

While creating cgroups, unpriv users can request a priv daemon to create BPF-based stats files.

Not liked by our container runtime team:

1. Overhead in the node management daemon.
2. Not highly available. Because the daemon may crash at any time, but cgroups can be created always (therefore stats files should be available always).

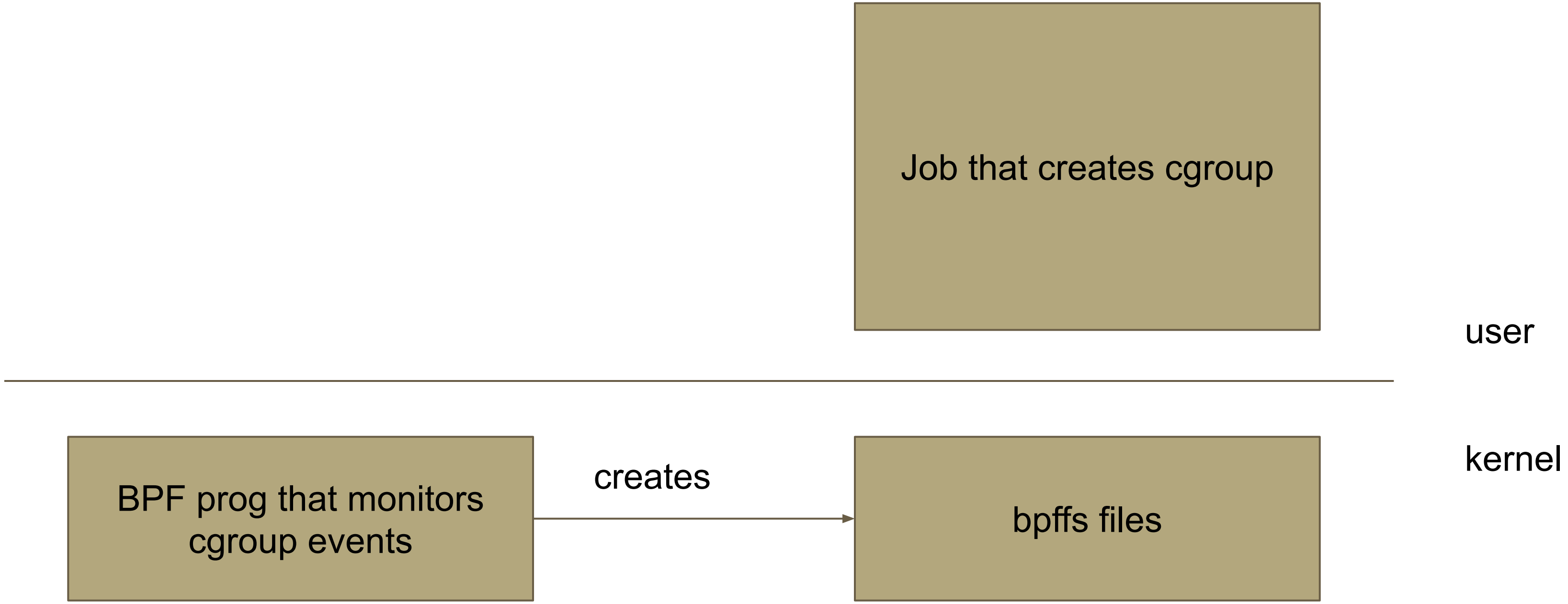

Hey, can you create a bpf stats for me?
Some privileged daemon
Job that creates cgroup
user
kernel
BPF iter prog
Attach + pin
bpffs files

Idea II

Deploy a BPF prog to monitor cgroup creation and let the prog to create the stats files. This is doable.

We need:

1. Sleepable tracepoint program type for monitoring cgroup events.
2. Special tracepoints in cgroup code.
3. Enable LINK_CREATE and OBJ_PIN bpf syscalls in sleepable TP BPF prog.


Job that creates cgroup
user
kernel
BPF prog that monitors cgroup events
creates
bpffs files

Related: Daemonless Container Engine

Daemonless is a recent concept in the container world.

Daemonless container engine spawns containers without the help of a priv daemon, which lowers the risk of jobs escaping the container.

Can we delegate more container runtime operations as BPF sleepable TP progs?

Idea III

Since the root problem is the privilege requirement for LINK_CREATE and OBJ_PIN, can we enable them for unpriv users?

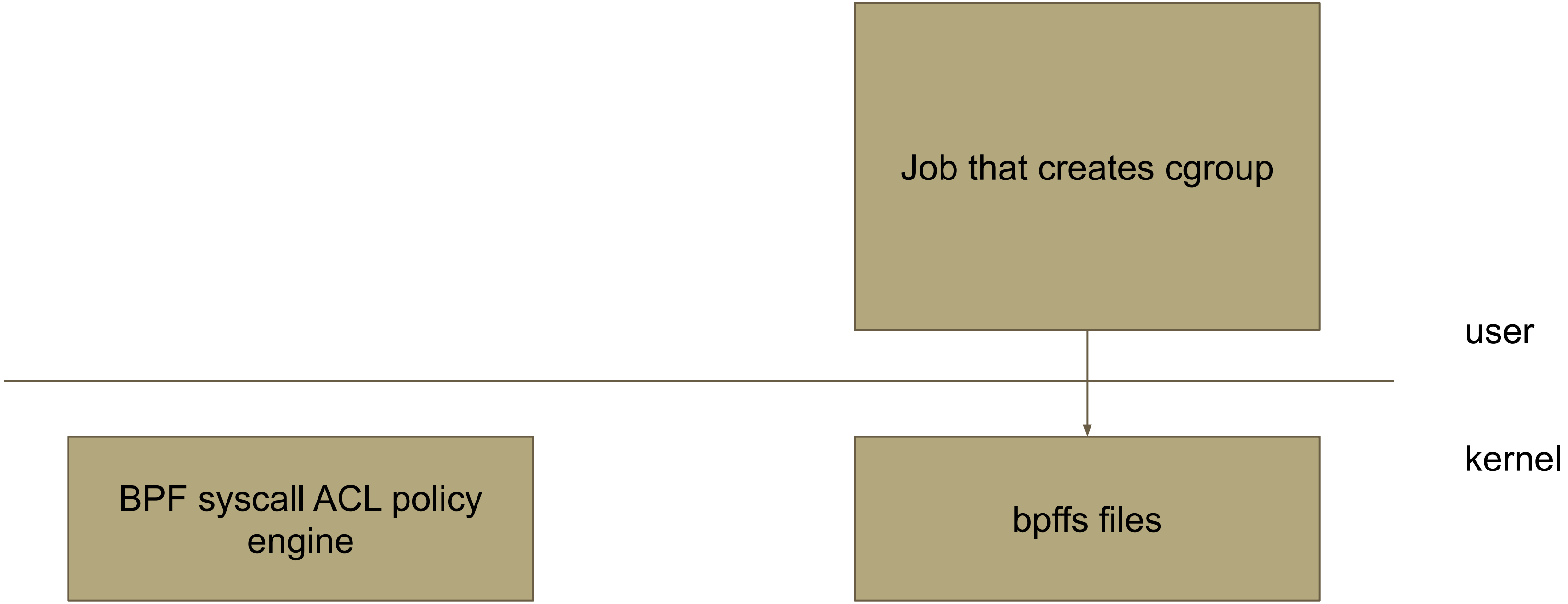
Fine-grained ACL for BPF Syscall Operations

Can we implement fine-grained ACLs for bpf operations?

More specifically, grant permission at a per-prog, per-operation basis.

For example, grant (cgroup stats iter, link_create) and (stats iter link, obj_pin) permission to unpriv users.

Maybe implement such a policy engine based on LSM bpf?


Job that creates cgroup
user
kernel
BPF syscall ACL policy engine
bpffs files

High Level Design - Collection

Cgroup stats properties

- Hierarchical: stats aggregated over cgroup subtree.

My colleague Yosry Ahmed is working on collection, “almost code complete, still testing”

Idea on Collection

rstat is an infrastructure built in the core of cgroup for collecting stats.

BPF-based collection is plugging BPF in rstat

- Introduce new prog type: *rstat flusher*
- Rstat flusher attaches to a cgroup subsystem and defines the logic for how to aggregate stats across CPUs and across hierarchy.

Status of BPF Colleciton

My colleague Yosry Ahmed is working on collection, “almost code complete, still testing”

The Last Bit: Map for storing cgroup results

We currently put the computed results in hashmap, but,

Can we change cgroup local storage to be generic like task local storage?

Cgroup local storage may be faster than hashmap in this case.

Thank You

Questions?