Device Plugin Allocate Invocation Call

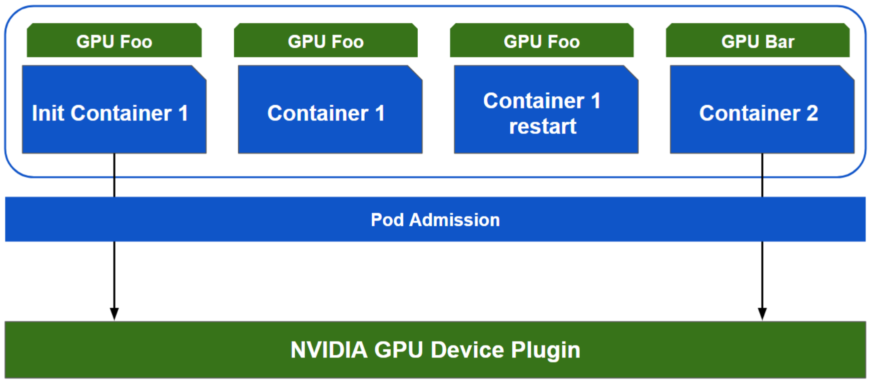
Author: @RenaudWasTaken <rgaubert@nvidia.com>

Introduction

In the device plugin system, the Allocate call is issued on a per "device group". It allows to execute vendor specific instructions before the container creation (e.g: scrub the GPU memory, cleanup GPU memory, reset device, setup an FPGA, ...) as well as communicate container runtime setup requirements to kubelet.

The current system issues one Allocate call per "device group" during pod admission, it caches the response and re-inject it to all containers that uses that device group.
However when paired with container restart and Init containers, reproducibility is not guaranteed leading to undefined behavior in case of container restart or device reuse between Init containers and containers.

Device Plugin Response: *contains runtime setup requirements that explain how to expose the device in a container. Currently this means we can send Environment variables, Devices, mounts and container annotations (intended to run a pre-start hook, which we would like to standardize in CRI) to Kubelet.*

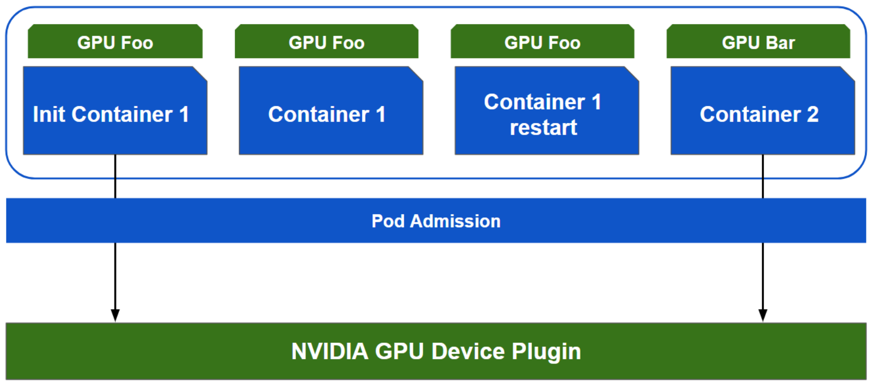


One Allocate call per device group
Response is cached and re-injected for every container using that device group

This document explores two solutions that are currently being discussed in PR #58282 as well as the current model:

- Allocate pod level
- Allocate per container
- Allocate per container with caching on restart

Allocate pod level



The current system issues one Allocate call per "device group" during pod admission. Meaning that for each container it will issue an Allocate call during pod admission, however if the device(s) are re-used exactly (i.e Init Container 1 = {devA, devB} and container 1 = {devA, devB}, but not Init Container 1 = {devA, devB} and container 1 = {devA}) it won't issue another allocate call.

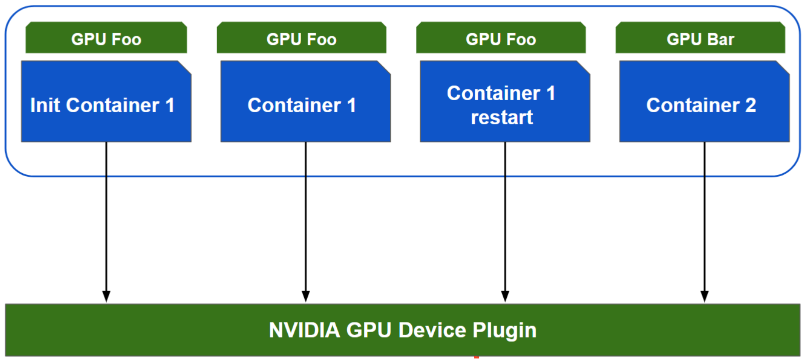
Pros:

- If the device plugin fails at any point during the pod lifecycle, it will have no impact on the different containers
- If the device plugin is not accessible during pod admission, the pod is rejected and re-scheduled to a different node

Cons:

- Only a very limited set of devices support that model
- Device reuse between Init containers and containers is essentially undefined behavior.
    - The init container might inadvertently change the device state thus changing the behavior of Container 1
    - If the device state change is on purpose, using the behavior of device re-use and not issuing the allocate call as a as a sharing mechanism to setup a device (e.g: using a Qantum Random Number Generator being in the init container to initialize it) isn't a good API. It's a side effect and an unreliable implicit API, instead an explicit API should be used (Resource Classes).
- Containers have an undefined behavior on restart. Restart usually means that something went wrong, and the container could have changed the device state or even put the device in a bad state.
    - NVIDIA has seen in the field memory not being cleaned up by the driver. We could issue a device reset to try and fix that behavior
- Device containers don't offer the same model or guarantees as normal containers
- Added complexity of checkpointing the device plugin response
- Added complexity of treating device Allocation call as a transaction that is not currently being handled

Allocate per container



Allocate gRPC call is done on [container start]

The device plugin was designed with executing vendor specific operations in mind in order to either setup a device (e.g: FPGA), perform security operations (e.g: Scrub the GPU memory) basically resetting the device in the same state for every container run. Thus keeping a basic container promise, reproducibility.

Such operations might be:

- Memory scrubbing
- Health check (e.g run a 1 minute job to verify GPU state) and try to reset it in case of bad state
    - Though this could be only during pod admission
- GPU Allocated memory checks
- "Zombie processes" checks
- ...

It would also be interesting to have feedback on the challenges faced by the Intel FPGA team, and teams working with VFIO and hypervisors who might have problematics imposing per-container Allocation.

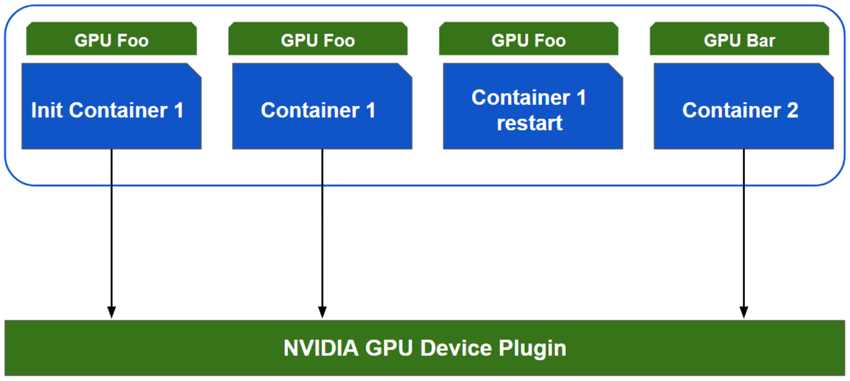
Pros:

- This allows for a more flexible model and thus support more devices
- Device containers offer the same guarantees as normal (i.e: cpu) containers

Cons:

- The device plugin might not be available during container restart
    - Though this is the same problem with CRI if it's gRPC remote is not available
- Container startup might take more time
    - There is a community discussion that needs to be had on what operations should be done at container start and the length of the operations
        - Though this is not new as mentioned by @Vishh and @derekwaynecarr plugins are intended to be a well-audited functionality.
    - We probably need to introduce a "pod admission call" or a parameter flagging the Allocate call as a container restart (i.e: Fast path, Slow path)

Allocate per container with caching



Allocate gRPC call is done on [container start]
Caching is performed to re-inject response on container restart

Allocate call would be done at container start but not on container restart.

Pros:

- This allows for a more flexible model and thus support more devices
- Restart would not incur the gRPC call and device specific operations

Cons:

- The device plugin might not be available during container start
    - Though this is the same problem with CRI if it’s gRPC remote is not available
- Container startup might take more time
- Container restart is still undefined