Accessing GPUs from Containers in HPC

Overview

1. Build an image capturing all application requirements
2. Push the image to DockerHub or a Private Registry
3. Launch the image as a Container

Docker drawbacks in HPC

- **Architecture** assumes a local disk
- **Security** users can easily escalate privileges on the host
- **Integration** not designed to work with batch systems
- **Complexity** uses a client/daemon architecture

Solution: Shifter

- **Flexibility** no administrator assistance to launch a container
- **Security** stripped-down version of image deployed in read-only mode
- **Integration** workload-manager integration, e.g., SLURM
- **Compatibility** full integration with public repositories, e.g., DockerHub

Shifter Architecture

- CUDA requires access to the kernel driver and the corresponding CUDA runtime libraries.
  1. Mount the character devices of the available GPUs (e.g., /dev/nvidia0).
  2. Mount the user-space libraries on the host system at deployment time.
  3. Reflect the mounted paths, e.g., using `ldconfig`, within the container runtime.

- The procedure is executed automatically as part of the workload-manager integration hooks, e.g., SLURM

GPU Support

- Containers are hardware- and platform-agnostic
- Support for GPUs requires:
  - specialized hardware, and
  - specific software on the host, e.g., Nvidia driver

Shifter approach

- **Flexibility** required libraries discovered at runtime
- **Security** access to character devices with user’s UID
- **Integration** direct access to character devices
- **Compatibility** works with any CUDA SDK v6.0 or later

Shifter Performance

- Seamless execution of CUDA- and OpenCL-enabled containers achieving native performance.
- Support for different CUDA toolkit versions on the host and on the container.
- Directly leverage GPU resources on a Cray XC30 like Piz Daint from within a container.

References