Make Accelerator Pluggable for Container Engine

www.huawei.com

Author/ Email: Jiuyue Ma / majiuyue@huawei.com

Version: V1.0 (20150504)

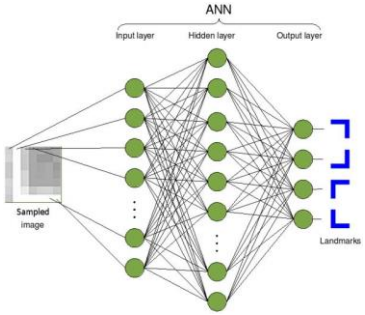
HUAWEI TECHNOLOGIES CO., LTD.

Today's Topic

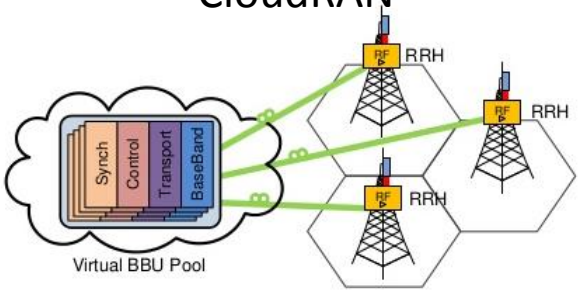
OLTP/OLAP

BigData Processing

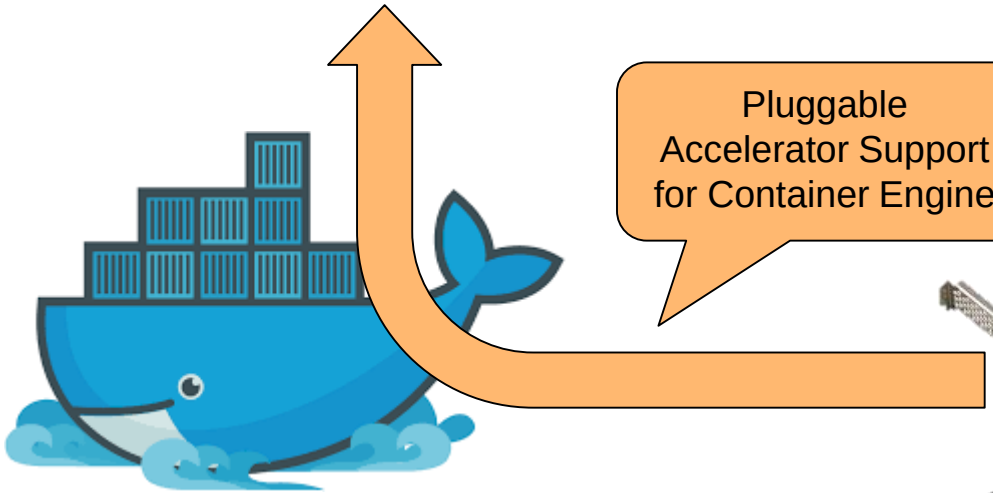
Neural Network



CloudRAN

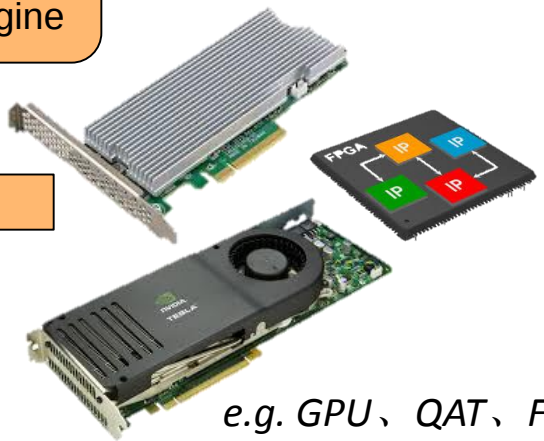


Pluggable Accelerator Support for Container Engine



Container Engine

Hardware Accelerators



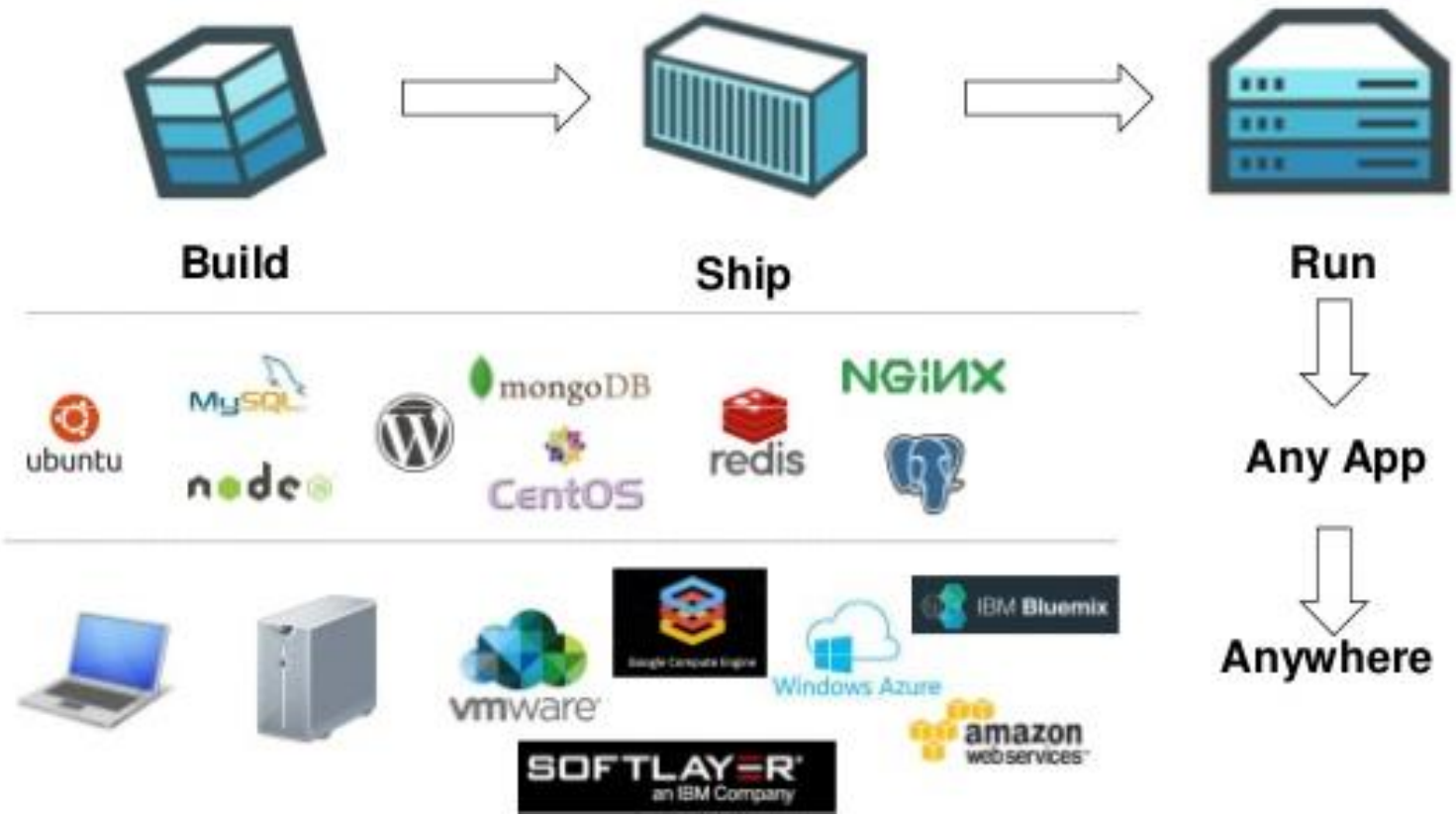
e.g. GPU、QAT、FPGA

Agenda

- Use accelerator in containers and How?
- Key problems to deal with
  - Identify accelerator requirements
  - Prepare accelerator runtimes
  - Manage accelerator resources
- Further development

Why Container?

~~Why Container?~~ Why NOT Container?



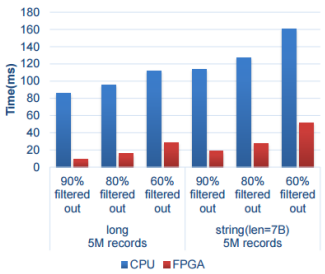
* Raphael Da Silva. OPEN SOURCE TECHNOLOGY: Docker Containers on IBM Bluemix. 2015.

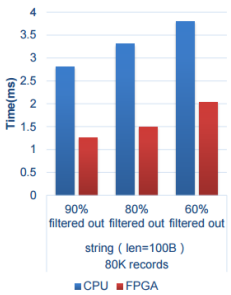
Why Accelerator?

- CPU is not fast/effective/… enough

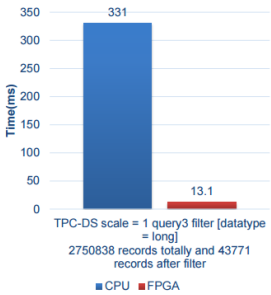
Evaluation - filter [baidu, HotChips 2016]

- Micro benchmark
  - At most 10x than dedicated C++ code

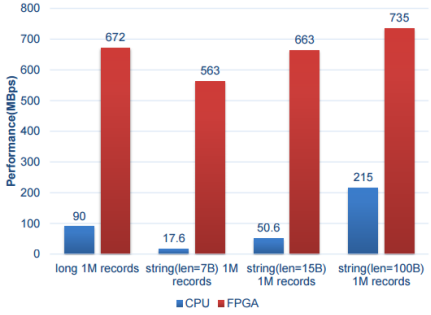




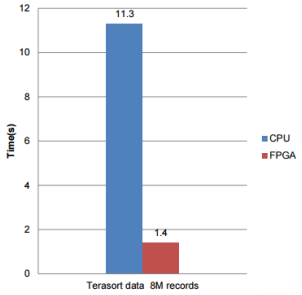
- Real case: TPC-DS query3
  - 25x faster than general C++ comparator



- Micro benchmark
  - At most 33x



- Terasort
  - 8x



- Heterogeneous is the future
  - Nvidia GPU-compute, Intel Skylake Xeon+FPGA, AWS P2/F1 Instance
  - use the right processor, in the right place, at the right time

Container a.k.a. Docker

Google it ! “Use XXX in docker”

Google

use DPDK in docker

DPDK

All Videos News Images Shopping More Settings Tools

About 50,800 results (0.70 secon

Can you run Intel's Data
https://developers.redhat.com/
Jun 2, 2015 - As part of our partic
are often involved in research anc

GitHub - redhat-performa
https://github.com/redhat-perfc
Contribute to **docker-dpdk** develc
Intel's Data-plane Development K

[PDF] Building High-Perfor
events.linuxfoundation.org/site
NIC. Driver. (VF). **DPDK**. *: http://
development-kit-**dpdk**-in-a- **docke**

6. Virtio_user for Contair
dpdk.org/doc/guides/howto/virt
6.2. Sample **Usage**. Compile **DPI**
Write a Dockerfile like below. cat
COPY . / Build a **Docker** image.
instance with a virtio-user port.
You visited this page on 5/3/17.

Can we run Intel DPDK i
https://groups.google.com/d/to
Oct 8, 2014 - just for cross-refere
container/231/3. On Wednesday,

Google

use FPGA in docker

FPGA

All Videos News Shopping Images More Settings Tools

About 16,400,000 results (0.39 seconds)

GitHub - intelsdi-x/fpga-
https://github.com/intelsdi-x/f
Bring **FPGA** accelerators as a res
one **using** su command (hence tl
You visited this page on 4/25/17.

[Proposal] Use accelera
https://github.com/docker/doc
Nov 20, 2016 - To **use** more hard
can be image labels or **docker ru**
You've visited this page 2 times.

GitHub - mithro/docker-
https://github.com/mithro/doc
Oct 20, 2014 - Package **Xilinx FP**
container with the following extra

GitHub - cdsteinkuehler,
https://github.com/cdsteinkue
... VMs and **Docker** containers) to
that adds the Debian packages re

The docker container | T
https://openfpgaduino.github.i
The goal of the **openfpgaduino** p
... The **docker** container. https://

openfpgaduino/fpga pu
https://hub.docker.com/r/oper
fpga. The **fpga** project for **openFI**
Repository. **OpenFPGAduino/fpg**

lizhizhou/cloudfpga - Do
https://hub.docker.com/r/lizhi
FROM lizhizhou/**openfpgaduino**
/usr/bin/nodejs /usr/bin/node E
You visited this page on 5/14/17.

Google

use GPU in docker

GPU

All Videos News Shopping Images More Settings Tools

About 2,250,000 results (0.62 seconds)

cuda - Using GPU from a docker container? - Stack Overflow
stackoverflow.com/questior
Aug 7, 2014 - Ok i finally mar
server 14.04 and i'm **using** th

NVIDIA’s GPU-Docker Solution

GitHub - NVIDIA/nvidia-docker: Build and run Docker containers ...
https://github.com/NVIDIA/nvidia-docker
nvidia-**docker** - Build and run **Docker** containers leveraging NVIDIA **GPUs**.
You've visited this page 5 times. Last visit: 4/30/17

how to use nvidia-docker in container when I want to use GPU · Issue ...
https://github.com/NVIDIA/nvidia-docker/issues/176
Aug 22, 2016 - Now I want to **use GPU in docker** container, I know I can install nvidia driver in container, but images has no portability , so I wat to use ...

GPU-Enabled Docker Container | NVIDIA
www.nvidia.com › ... › Data Center Management Tools
Docker, the leading container platform, can now be **used** to containerize **GPU**-accelerated applications. This means you can easily containerize and isolate ...

NVIDIA Docker: GPU Server Application Deployment Made Easy
https://devblogs.nvidia.com/.../nvidia-docker-gpu-server-application-deployment-ma...
Jun 27, 2016 - nvidia-**docker** is essentially a wrapper around the **docker** command that transparently provisions a container with the necessary components to execute code on the **GPU**. It is only absolutely necessary when **using** nvidia-**docker** run to execute a container that **uses GPUs**.

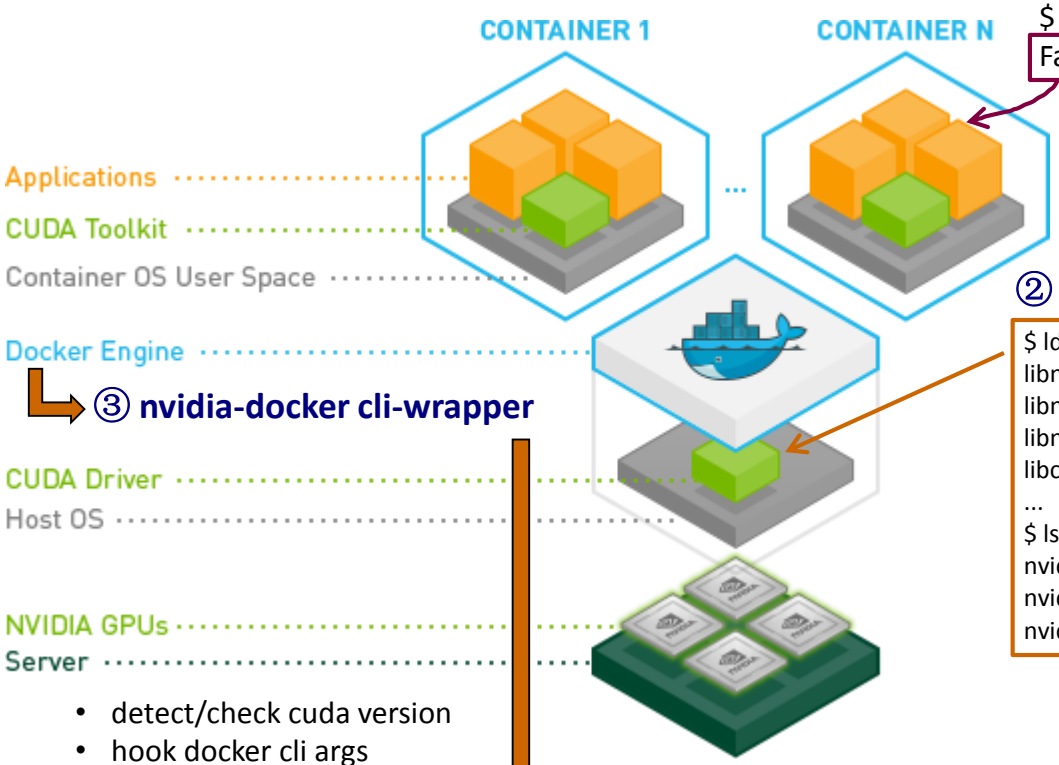
Nvidia plugin makes GPU acceleration possible in Docker containers ...
www.infoworld.com/.../nvidia-plugin-makes-gpu-acceleration-possible-in-docker-con...
Jun 29, 2016 - The **Docker** images that **use** the **GPU** have to be built against Nvidia's CUDA toolkit,

Use GPU in Docker (from nvidia-docker)

Key Problem： **VERSION MISMATCH**

$ nvidia-smi

Failed to initialize NVML: Driver/library version mismatch



① Labeled Image

```
LABEL com.nvidia.volumes.needed="nvidia_driver"
LABEL com.nvidia.cuda.version="7.5"
```

② VOLUME: nvidia-docker/nvidia_driver_361.48

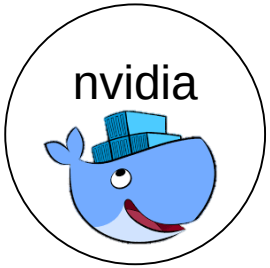
```
$ ldconfig -p | grep -E 'nvidia|cuda'
libnvidia-ml.so (libc6,x86-64) => /usr/lib/nvidia-361/libnvidia-ml.so
libnvidia-glcore.so.361.48 (libc6,x86-64) => /usr/lib/nvidia-361/lib…
libnvidia-compiler.so.361.48 (libc6,x86-64) => /usr/lib/nvidia-361/lib…
libcuda.so (libc6,x86-64) => /usr/lib/x86_64-linux-gnu/libcuda.so
…
$ lsmod | grep nvidia
nvidia_uvm          711531  2
nvidia_modeset      742329  0
nvidia             10058469  80 nvidia_modeset,nvidia_uvm
```

③ nvidia-docker cli-wrapper

- detect/check cuda version
- hook docker cli args
  --volume-driver / --volume

Wrapped Command

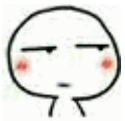
```
docker run
   --volume-driver=nvidia-docker
   --volume=nvidia_driver_361.48:/usr/local/nvidia:ro
```

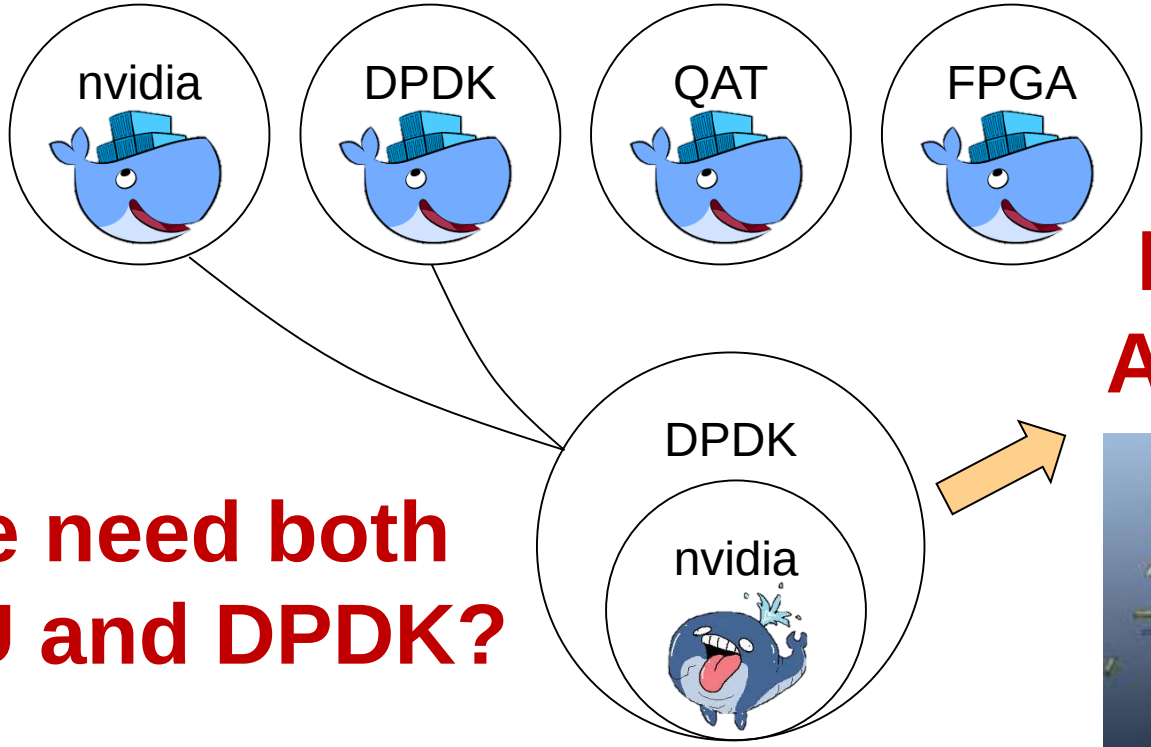


Do we really need “nvidia-docker”?

If so, we also need:

- dpdk-docker, qat-docker, fpga-docker, …





Pluggable Accelerator

But,
if we need both GPU and DPDK?

Pluggable Accelerator, How?

- What's the difference?

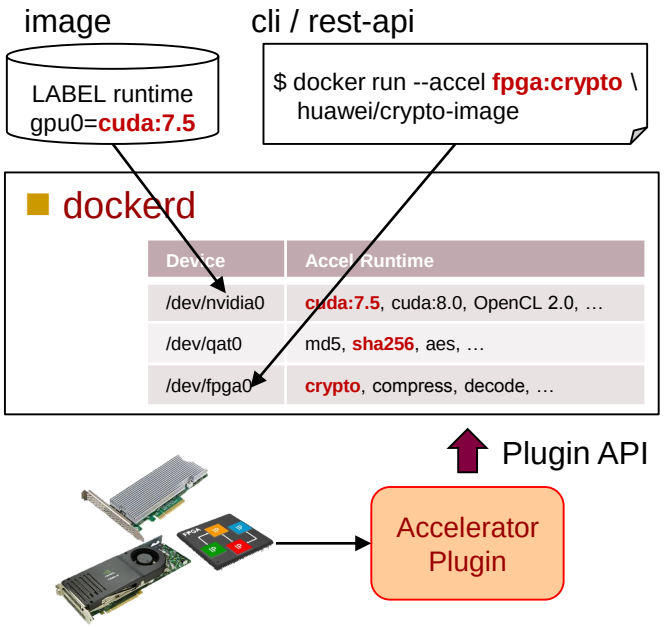
in-house FPGA Accel

	nvidia	DPDK	FPGA
Device Files (-d device)	/dev/nvidia0 /dev/nvidia-smi /dev/nvidia-uvm	/dev/uio0	/dev/fpga0
File Bindings (-v src:dest)	/usr/lib/nvidia-361/libnvidia-ml.so /usr/lib/nvidia-361/libnvidia-glcore.so.361.48 /usr/lib/nvidia-361/libnvidia-compiler.so.361.48 /usr/lib/x86_64-linux-gnu/libcuda.so ...	/sys/bus/pci/devices/xxxx:xx:xx.x /dev/hugepages	/usr/lib/libfpga.so /sys/bus/pci/devices/xxxx:xx:xx.x /dev/hugepages
Environments (-e env)	PATH=$PATH:/usr/lib/nvidia-361/bin LD_LIBRARY_PATH=...		

Accelerator Plugin Interface

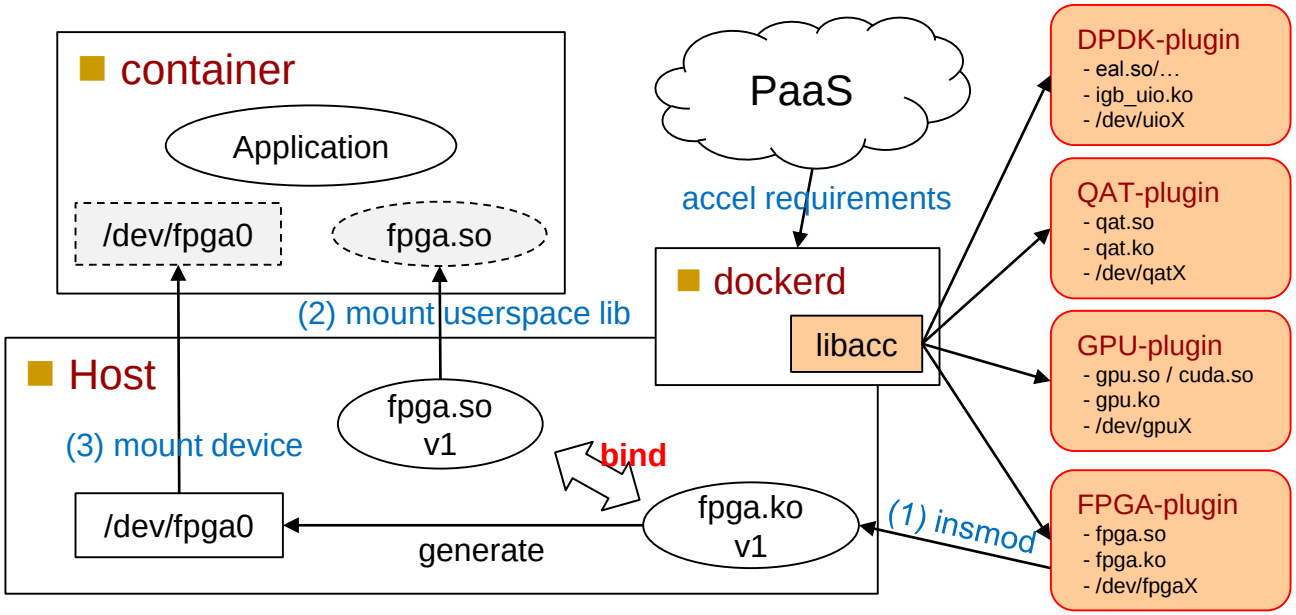
Implementation (1/3): IDENTIFY

- Accelerator Requirements
  - We are using the function provided by device, not the device itself
  - Identified by **runtime** instead of device
    - e.g. “A SHA256 accelerator” instead of “A FPGA with XXXX-bitstream”
- Identify user requirements
  - Integrate into image
  - Specify when start container
- Identify device capabilities
  - Identified by “accelerator plugin”
  - Report to engine through plugin api



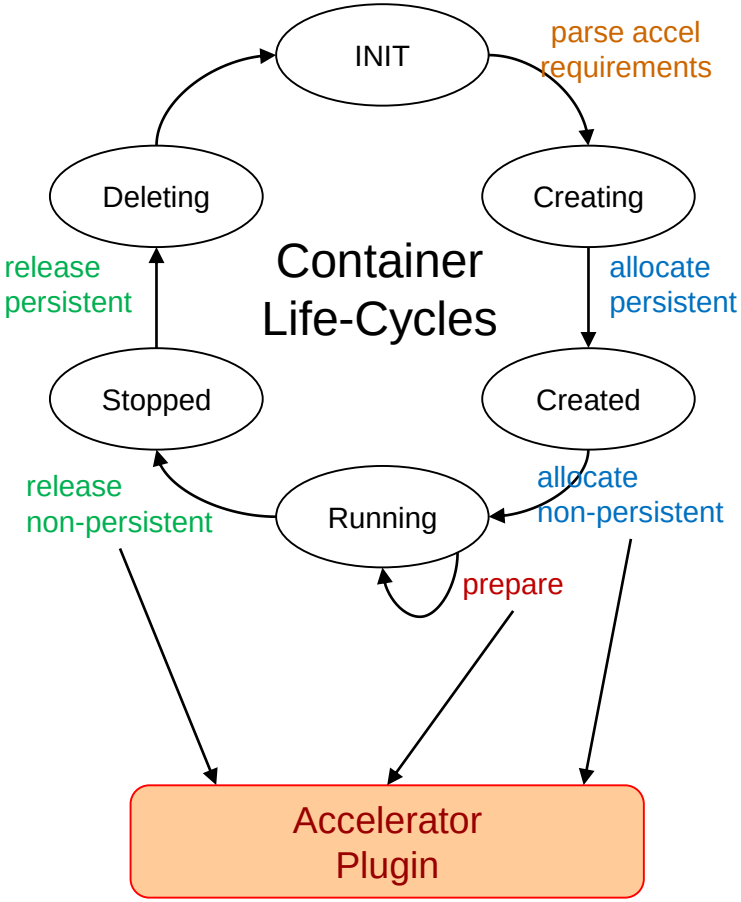
Implementation (2/3): PREPARE

- Select plugin from accelerator requirements
- Device initialization by plugin
- Dynamic inject required libraries & devices



Implementation (3/3): MANAGEMENT

- Accelerator Life-Cycle
  - Allocate → Prepare → Release
  - Plugin hooks for each stage
- Integrate with Container Life-Cycle
  - Persistent: Create→Delete
  - Non-persistent: Start→Stop



Example Flow

docker run -ti
--accel **snow3g@fakefpga, device=0000:06:04.4**
busybox bash

POST /container/create

```
{
  HostConfig: {
    Accelerators:[{
      Name: "anon_cli_accel_0",
      Runtime: "snow3g",
      Driver: "fakefpga",
      Options: ["device=0000:06:04.4"],
      IsPersistent:true,
      Sid: ""
    }]
  }
}
```

POST /container/xxxx/start

docker cli → docker daemon → containerd → runc

POST /container/create

- Daemon.ContainerCreate()
  - Daemon.create()

```
// merge accelerator runtimes required by image with cli specified runtime
// into contianer HostConfig, and verify these runtime requirements can
// be satisfied
if err := daemon.mergeAndVerifyAccelRuntime(params.HostConfig, img); err != nil {
    return nil, err
}
```

```
// allocate resources for "persistent" accelerators
if err := daemon.allocatePersistentAccelResources(container); err != nil {
    return nil, err
}
```

/var/lib/docker/containers/xxxxx/**config.v2.json**
/var/lib/docker/containers/xxxxx/**hostconfig.json**

POST /container/xxx/start

- Daemon.ContainerStart()
  - Daemon.containerStart()

```
if err := daemon.initializeAccelResources(container); err != nil {
    return err
}
```

  - Daemon.createSpec()

Binding

```
mounts = append(mounts, c.IpcMounts()...)
mounts = append(mounts, c.TmpfsMounts()...)
mounts = append(mounts, accelMounts(c.HostConfig.AccelBindings)...)
if err := setMounts(daemon, &s, c, mounts); err != nil {
    return nil, fmt.Errorf("linux mounts: %v", err)
}
```

Device

```
for _, deviceMapping := range append(c.HostConfig.Devices, getAccelDevices(c.HostConfig.Acc
    d, dPermissions, err := getDevicesFromPath(deviceMapping)
    if err != nil {
        return err
    }
    devs = append(devs, d...)
    devPermissions = append(devPermissions, dPermissions...)
}
```

Env

```
if len(c.HostConfig.AccelEnvironments) > 0 {
    s.Process.Env = daemon.mergeEnv(s.Process.Env, c.HostConfig.AccelEnvironments)
}
```

config.json

```
"process": {
  "env": [
    "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:
            /bin:/usr/local/cloud_acc_vf_driver/bin",
    "HOSTNAME=7f4c0c840861",
    "TERM=xterm",
    "LD_LIBRARY_PATH=/usr/local/cloud_acc_vf_driver/lib"
  ],
},
"mounts": [
  {
    "destination": "/usr/local/cloud_acc_vf_driver",
    "type": "bind",
    "source": "/var/lib/cloud_acc_vf_driver",
    "options": [
      "rbind",
      "ro",
      "rprivate"
    ]
  }
],
"linux": {
  "devices": [
    {
      "path": "/dev/cloud_acc_vf",
      "type": "c",
      "major": 768,
      "minor": 1635,
      "fileMode": 8576,
      "uid": 0,
      "gid": 0
    }
  ],
},
```

Further Developments

- Standardize “Accelerator Runtime”
  - OCI Runtime/Image Spec should be good place
- Integrate with Docker Swarm / K8S / etc
  - Cluster level accelerator schedule&management
- More Features
  - Accelerator Sharing
  - Accelerator Exception Interface
  - Accelerator Hot-Plug
  - …

Thank you

www.huawei.com