ML inference acceleration on k8s

> with kata containers & AWS Firecracker

nubificus

Containers devroom FOSDEM'21

Orestis Lagkas Nikolos✢, Babis Chalios, Anastassios Nanos

✢ Computing Systems Lab @ National Technical University of Athens

https://github.com/nubificus
@nubificus
https://blog.cloudkernels.net
https://nubificus.co.uk
info@nubificus.co.uk

501 West One Peak, 15 Cavendish street,
S3 7SR Sheffield, UK
Registered in England and Wales, #11545167

Problem Definition

Sharing accelerator devices on multi-tenant environments is an open problem

- Hardware partitioning (container/VM pass-through)
- API remoting
- Paravirtual drivers (NVIDIA vGPU)

What happens with serverless?

- How do we expose acceleration capabilities in a safe fashion to functions?

Hardware acceleration in the Cloud & at the Edge

nubificus

Hardware partitioning:

- bound to hardware vendor/device support
- disallow flexible sharing of diverse accelerator resources

API remoting:

- still either device/vendor API specific, or
- can incur significant performance overhead
- not fit for infrastructures with resource or performance (i.e. latency) constraints

Paravirtualization:

- users have to program the hardware directly
- multiple schedulers doing the same job (VM, VMM, runtime system)
- software stack duplication

Workload acceleration made simple: vAccel

vAccel semantically exposes "accelerate"-able functions to users, while supporting a wide range of acceleration frameworks.

Design goals:

- programmability / simplicity
- performance (minimal overhead)
- portability / interoperability
- virtualization support

vAccel: architectural overview

nubificus

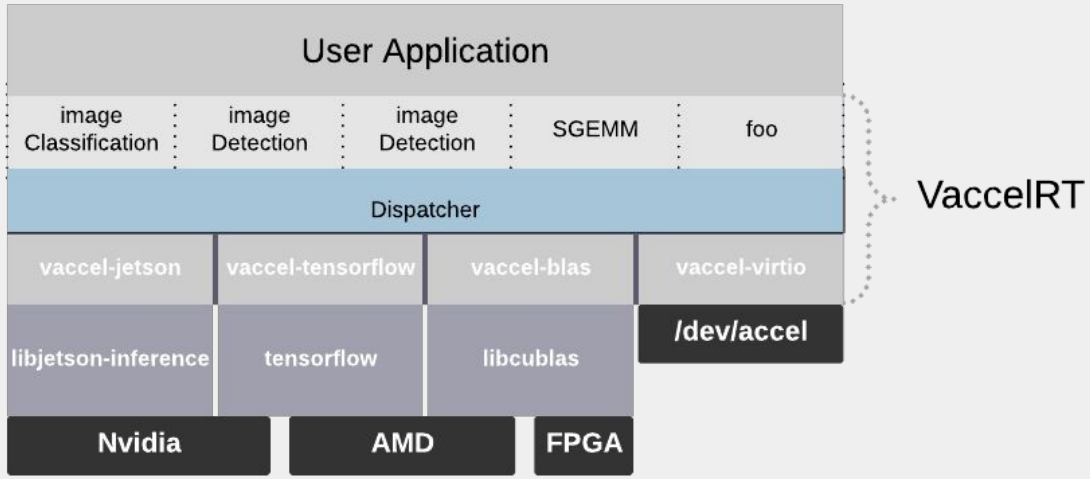
core component: **vAccelRT (vAccel runtime system)**

user-facing API: **function prototypes**

- **abstracted by the underlying frameworks or**
- **defined by the system as a superset / subset of individual acceleration functions**

hardware abstraction layer: **acceleration frameworks, transport layer**

- **low-level APIs (openCL, CUDA, openACC etc.)**
- **higher-level frameworks (TensorRT, tensorflow, pytorch etc.)**
- **user-facing APIs (jetson-inference, libBLAS etc.)**
- **virtio-accel**



vAccel: implementation

nubificus

current PoC: QEMU/KVM, AWS Firecracker

- vAccelRT written in C runs on recent Linux distr.
  - Bindings for Rust
- PV:
  - virtio-accel written in C as a Linux Kernel module (>5.4)
  - virtio-accel backend written in C for QEMU, Rust for AWS Firecracker
- backends:
  - support OpenCL, CUDA, tensorRT and jetson-inference

vAccel: orchestration

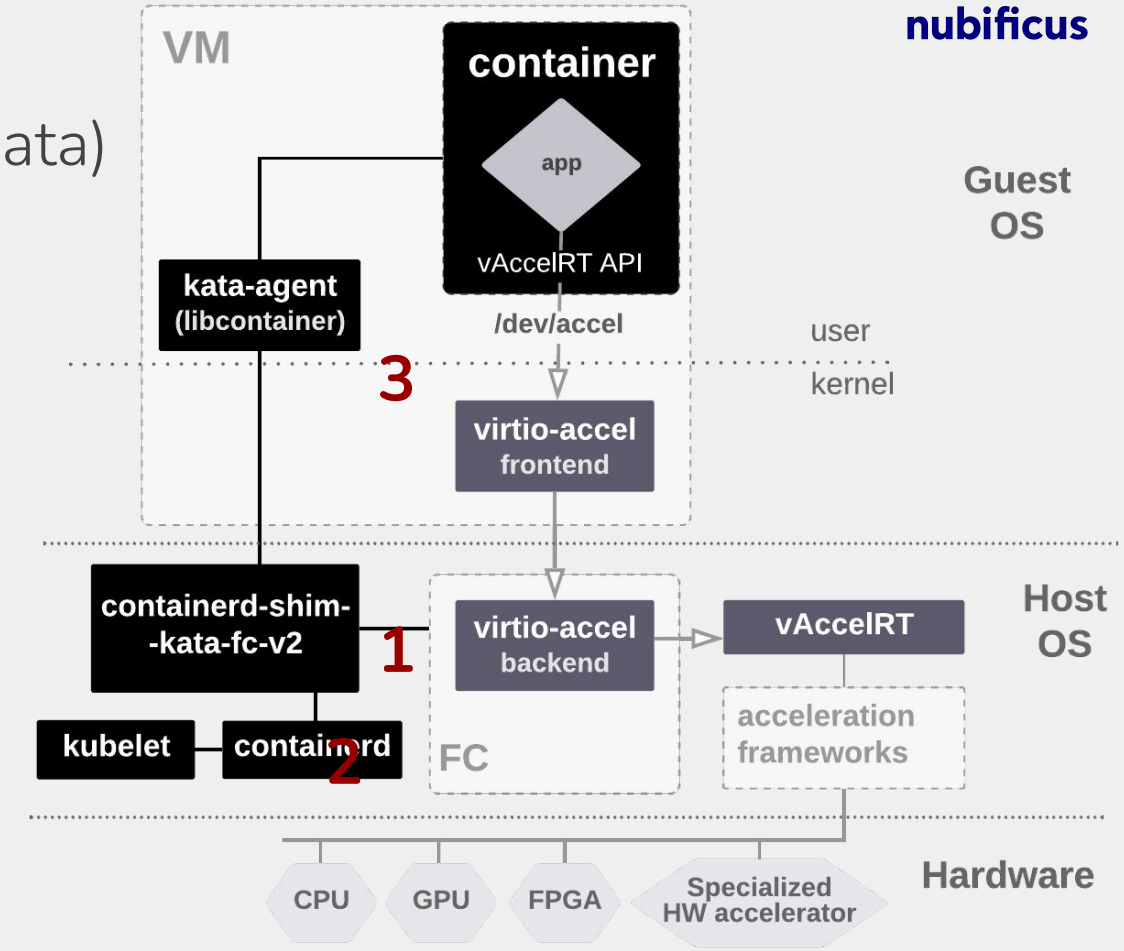
AWS Firecracker with vAccel on k8s (using kata)

1. kata v2 runtime (1.12 & 2.0) with Firecracker v0.21.1
2. docker
3. custom device assignment

vAccel: orchestration

AWS Firecracker with vAccel on k8s (using kata)

1. kata v2 runtime Firecracker <= v0.21.1
   -> **v0.23.1**
2. docker with devmapper
   -> **containerd (still using devmapper)**
3. custom device assignment
   -> **kata-agent patch**



vAccel: k8s installation process

nubificus

1. install kata-containers on k8s (RuntimeClass, containerd conf)

2. label k8s nodes with accelerators support (e.g vaccel=”true”)

3. install vAccel binaries on “vaccel” labeled nodes
   (Firecracker, kernel, rootfs image & config)

4. ensure cuda/acceleration frameworks are installed on the labeled nodes and working (nvidia-smi)

vAccel: demo

nubificus

vAccel: Give it a try!

nubificus

https://vaccel.org/

- Installation guide for k8s
- simplified on k3s

Thanks!

https://github.com/nubificus
@nubificus
https://blog.cloudkernels.net
https://nubificus.co.uk
info@nubificus.co.uk

501 West One Peak, 15 Cavendish street,
S3 7SR Sheffield, UK
Registered in England and Wales, #11545167

vAccel in FOSDEM’21

ML inference acceleration for lightweight VMMs

Where: Virtualization & IaaS devroom

When: 2021-02-06 | 12:15:00

Hardware acceleration for unikernels

Where: Microkernels devroom

When: 2021-02-06 | 13:45:00