ML inference acceleration for lightweight VMMs

Virtualization and IaaS devroom FOSDEM'21

nubificus

Anastassios Nanos, Babis Chalios, Kostis Papazafeiropoulos✢

✢ 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

Overview

- Problem
- Hardware acceleration landscape
- Serverless
- vAccel core concept & design
- Implementation walkthrough
- Demo & Roadmap

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 about serverless?

- How do we expose acceleration capabilities, securely, to functions?

Hardware acceleration in the Cloud & at the Edge

Hardware partitioning

- bound to hardware device/vendor support
- inflexible 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

The Serverless use-case

Hardware acceleration support is currently missing from popular Serverless platforms

Considerations for such a platform:

- **Security:** users should be able to share hardware resources securely
- **Hardware abstraction:** users should not deal with vendor/device-specific APIs
- **Programmability/Portability:** users should enjoy an intuitive programming interface for accessing hardware acceleration (+ideally vendor-agnostic).

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 (device/vendor-agnostic)
- performance (minimal overhead)
- portability / interoperability (run anywhere)
- security / isolation (virtualization support)

vAccel: architectural overview

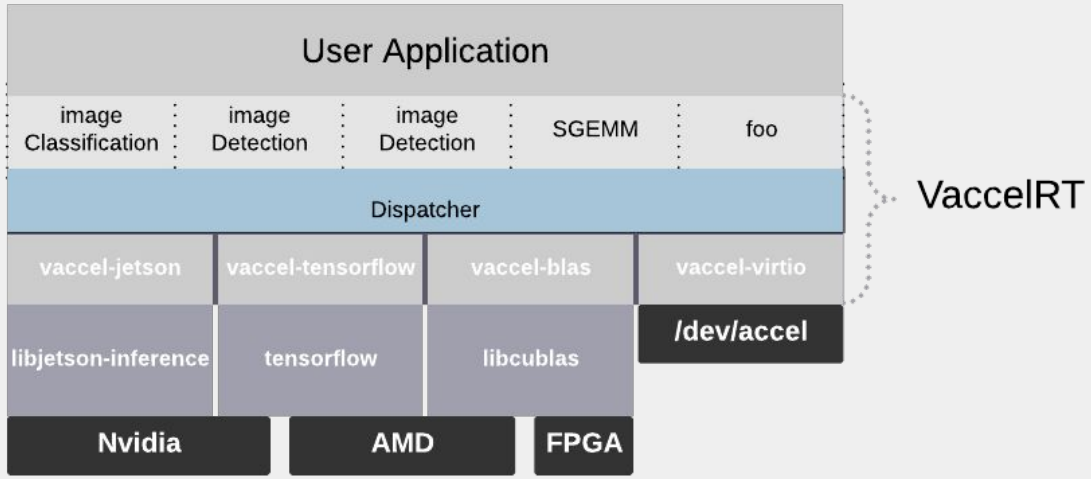
core component: **vAccelRT (vAccel runtime system)**

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

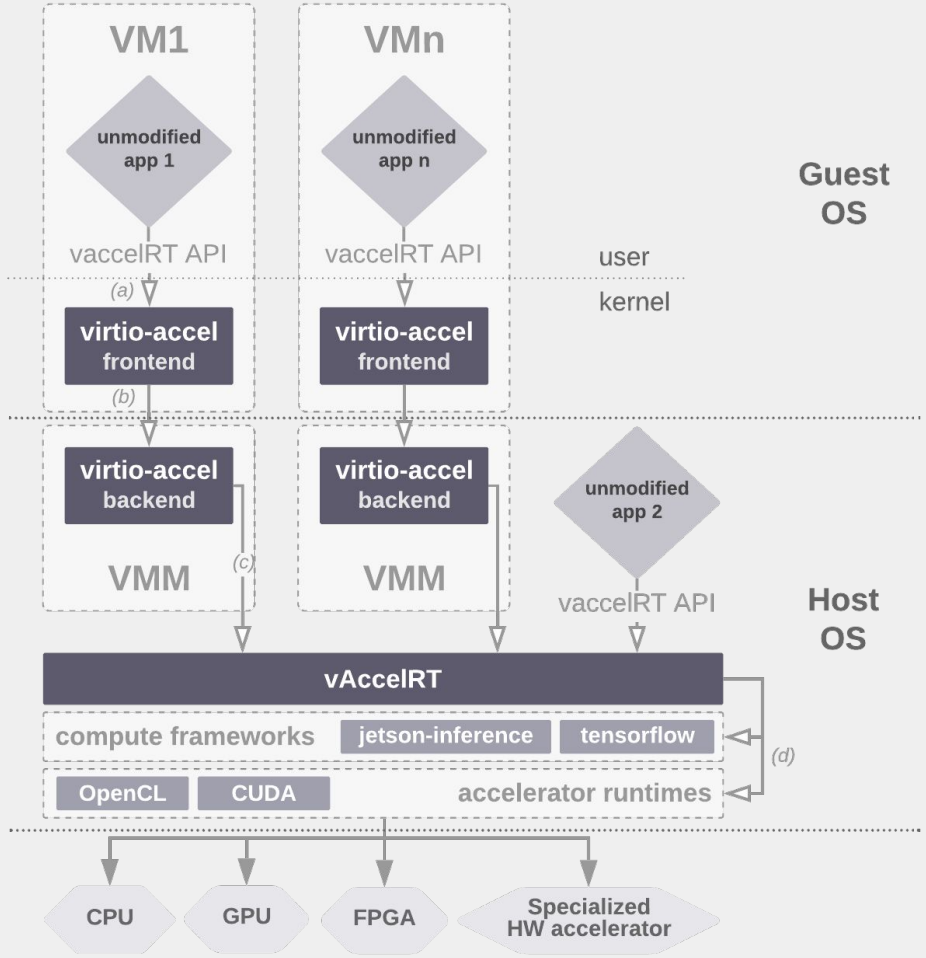
current PoC: QEMU/KVM, AWS Firecracker

- vAccelRT written in C runs on recent Linux distro
  - Bindings for Rust
- backends:
  - Currently: support for jetson-inference, OpenCL and the virtio-accel plugin
- 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

vAccel: example execution flow for a VM

application

```
image_classify(image, model, parameters, &output)
```



vAccel: example execution flow for a VM

application

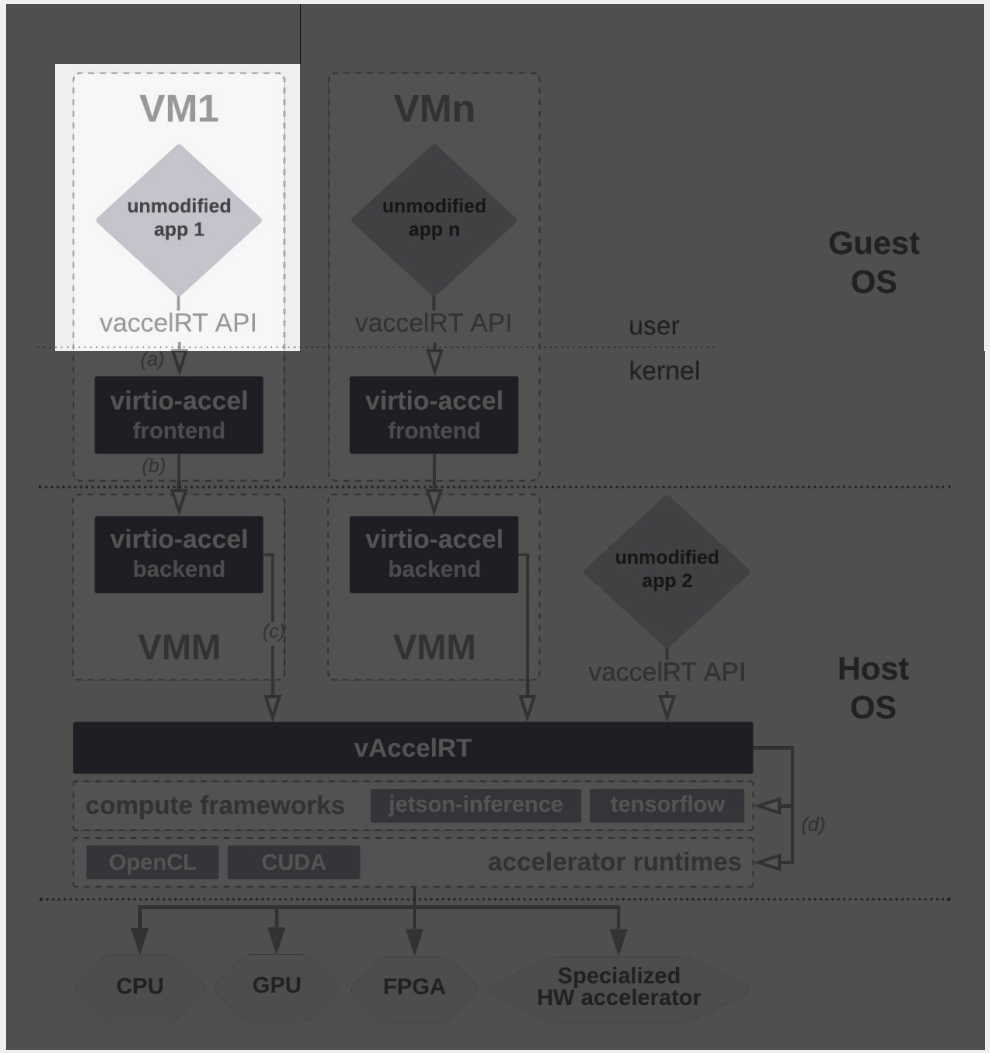
```
image_classify(image, model, parameters, &output)
```

VaccelRT (inside the guest)

1. Look for backend plugin that implements image classification
2. If supported, call the plugin code

vaccel-virtio plugin

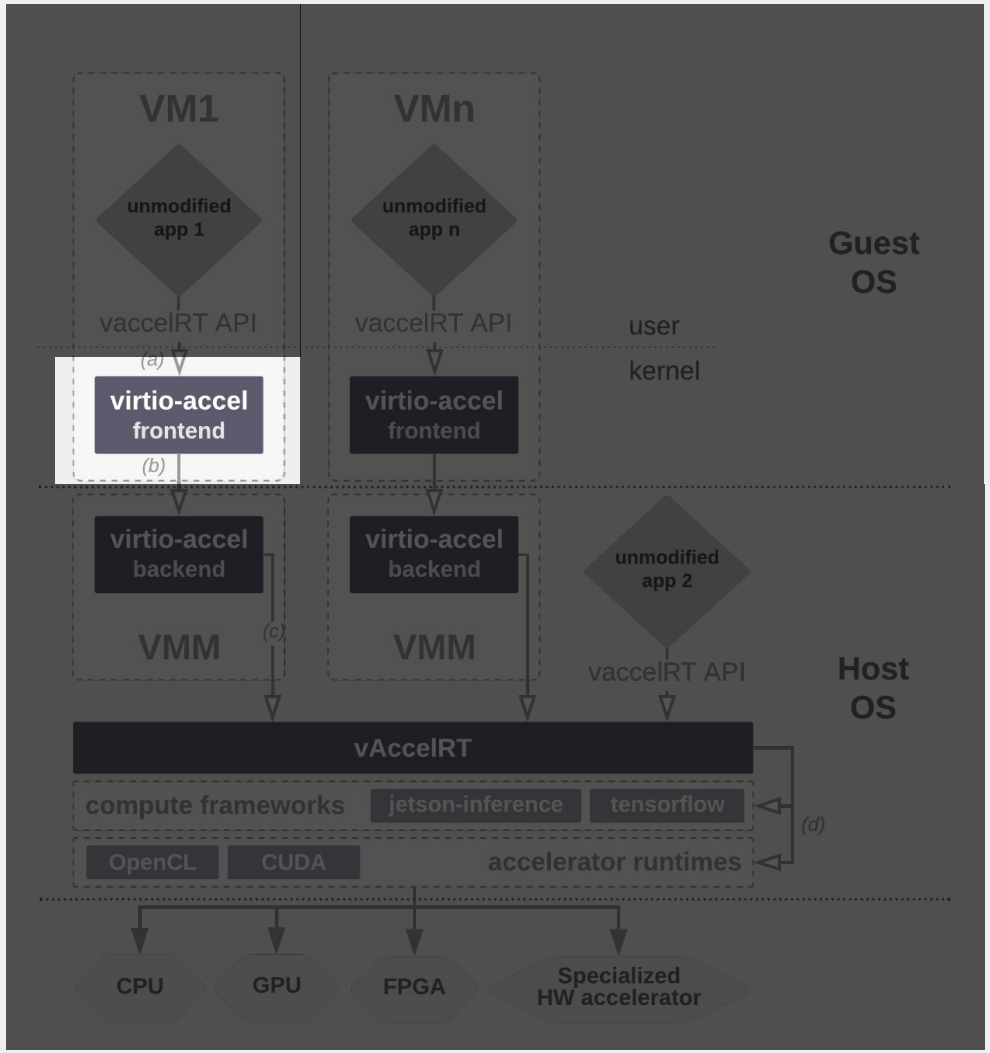
1. Check that the virtio device is present
2. Prepare arguments and issue `ioctl` command to the vaccel device



vAccel: example execution flow for a VM

vaccel-virtio front-end driver

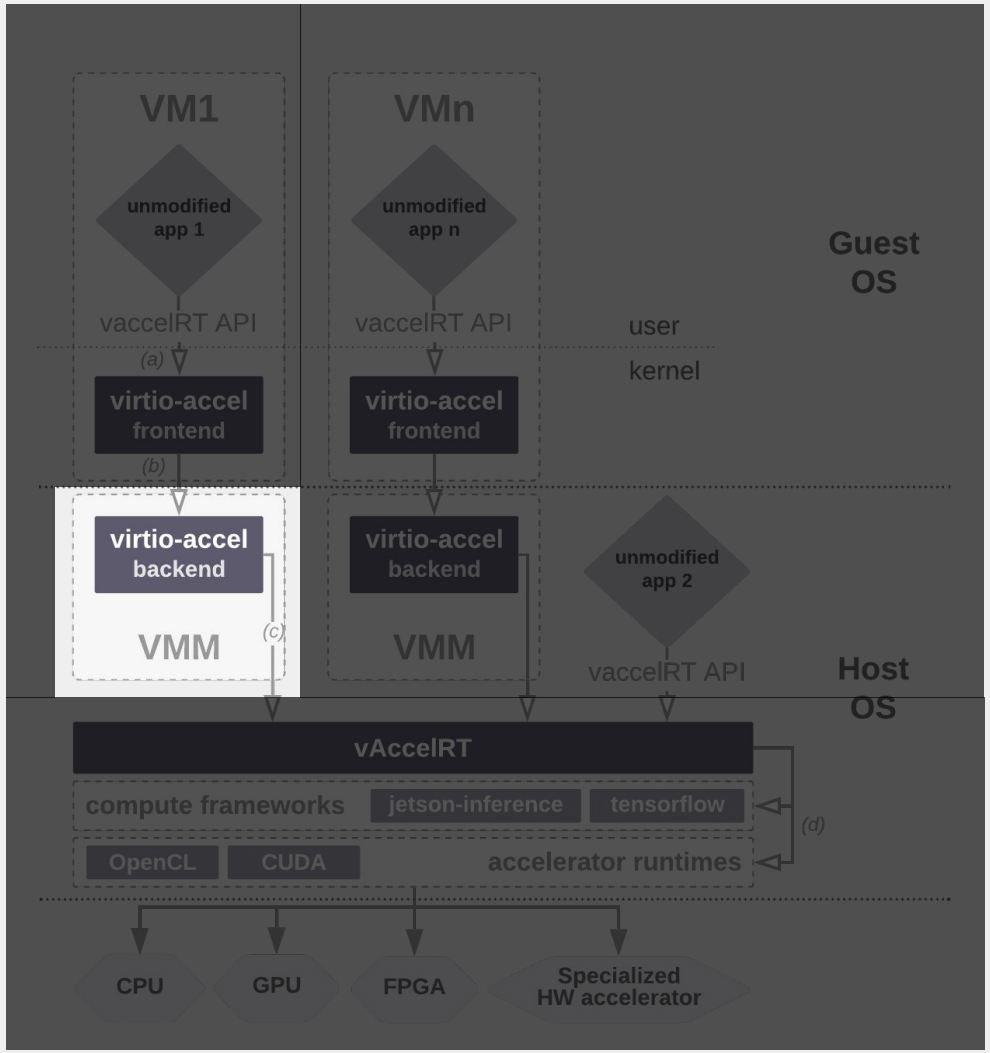
1. Create virtio request
2. Insert user arguments inside the request
3. Insert the request inside the virtqueue
4. Kick the virtqueue



vAccel: example execution flow for a VM

VMM

1. The guest VM exits in the VMM
2. The virtio-backend inside the VMM
   a. Parses the request
   b. Validates the request header and corresponding arguments
   c. `image_classify(image, model, parameters, &output)`



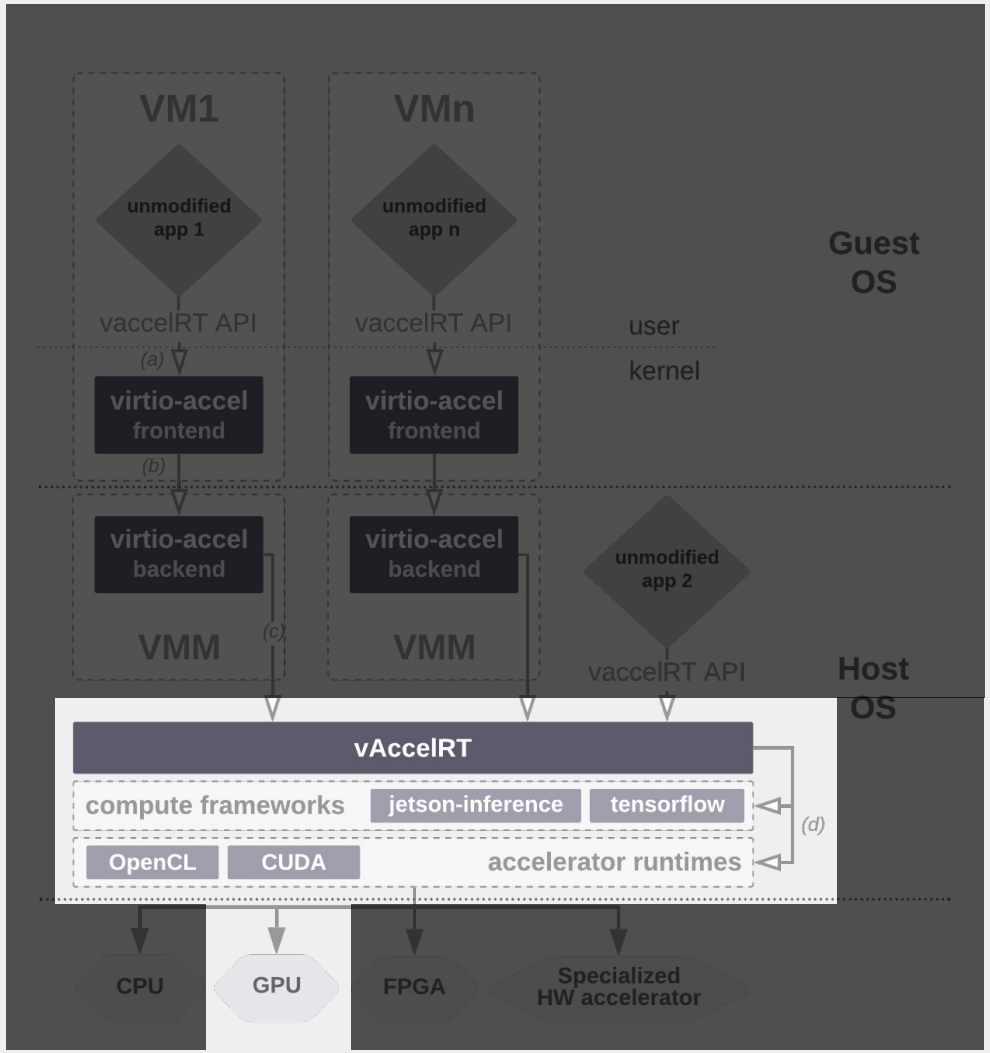
vAccel: example execution flow for a VM

VaccelRT (in the host)

1. Look for the backend plugin that implements image classification
2. If supported, call the plugin code

vaccel-jetson plugin

1. Perform operation using jetson-inference framework
2. Offload computation to GPU



Future Steps

1. Stabilize the user-facing API
2. Develop more back-end plugins
   a. Target more accelerator devices
   b. Provide more accelerated functions
3. Investigate vsock as a transport layer
   a. Custom virtio-accel avoids potential network-stack overhead
   b. vsock will allow us to decouple changes in the API from the VMM & use vAccel with any VMM that implements vsock
4. Security oriented features
   a. Provide stronger guarantees with regards to user state residing in the accelerator
   b. Allow only certified calls to be handled in the host
5. Provide bindings for more languages
   a. Python, Go, NodeJS, etc.

vAccel: demo

Summary

merits

- user-friendly / code portability:
  - issue calls to generic functions
  - if hw acceleration is available then we get optimal results -- if not, then function will still run but on cpu.
- no direct access to hardware:
  - there is no way for the user to access gpu memory directly
- flexibility:
  - session-based execution, easily migrate-able to a different host with or without the same hardware.

limitations

- glue code to support backends
  - low-level code needs to be written for higher-level functions
- security implications:
  - libnvidia/cudart/tensorflow use a number of syscalls that should be whitelisted in seccomp

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 on k8s using kata containers & AWS Firecracker

Where: Containers devroom

When: 2021-02-07 | 17:05:00

Hardware acceleration for unikernels

Where: Microkernels devroom

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

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement no 871900 (5G-COMPLETE)

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