# Hardware acceleration for unikernels

nubificus

**Microkernel devroom** **FOSDEM'21**


Charalampos Mainas, Anastassios Nanos, Babis Chalios, Konstantinos 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

# What are we gonna talk about

nubificus

- Motivation
- Problem definition
- vAccel
- Serverless/Unikernels
- vAccel on Unikernels

# Current state


Modern applications

- IoT
- Mobile computing
- Web services

*What about applications whith large scale computations?*

# 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

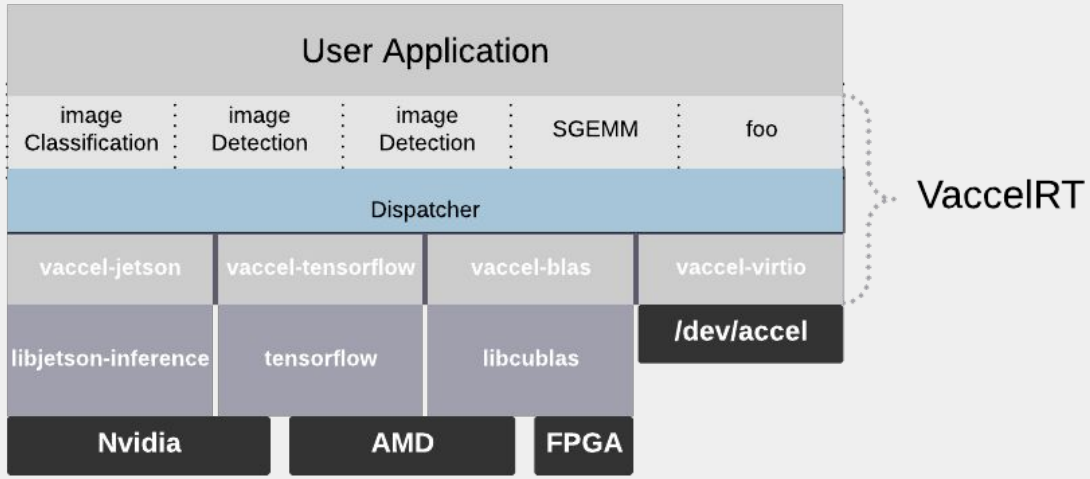
core component: **vAccelRT (vAccel runtime system)**

frontend: **function prototypes**

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

backend: **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: example execution flow for a VM

application calls

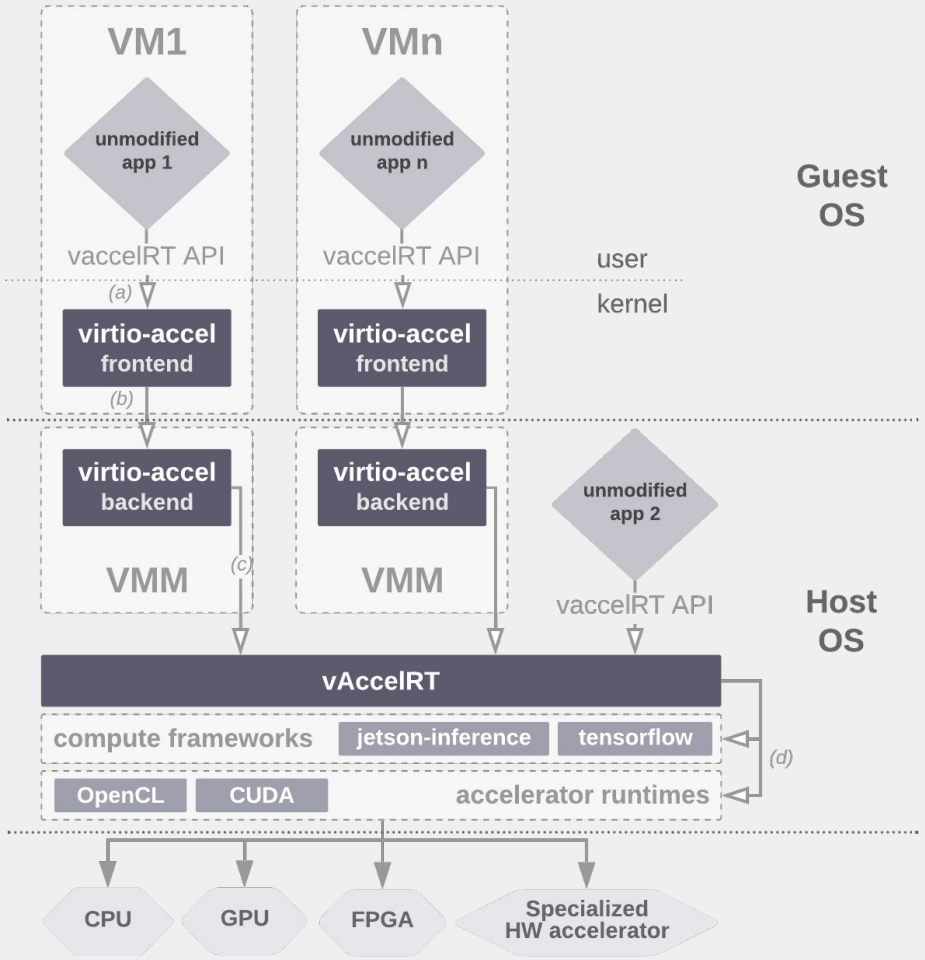
```
image_classify(image, model, parameters, &output)
```

a. vAccelRT determines available backends, chooses what's sane to do (use virtio-backend, use physical device etc.)

b. use virtio-accel, call forwarded to virtio-accel backend

c. virtio-accel calls vAccelRT

d. vAccelRT determines acceleration framework and issues the relevant call(s) to be offloaded to the hardware. Returns results to vAccelRT, which, forwards the output to the caller.



# VMs & Serverless


- Using VMs seem like an overkill...
- Containers
- Are there any other solutions?

- What about unikernels

# Unikernels for Serverless

nubificus

- ultra-fast spawning
- Security (isolation, reduced attack surface)
- minimal memory footprint overhead
- highly scalable (scale out)

still... what about applications with demand on hardware acceleration

# Unikernels for Serverless with HW accel

nubificus

**Introducing hardware acceleration in unikernels with vAccel**

- Ideal abstraction
- Easy to port (no hw specific code, only semantic abstractions)
- Easy to use it
- Small overhead

# vAccel: implementation on unikernels

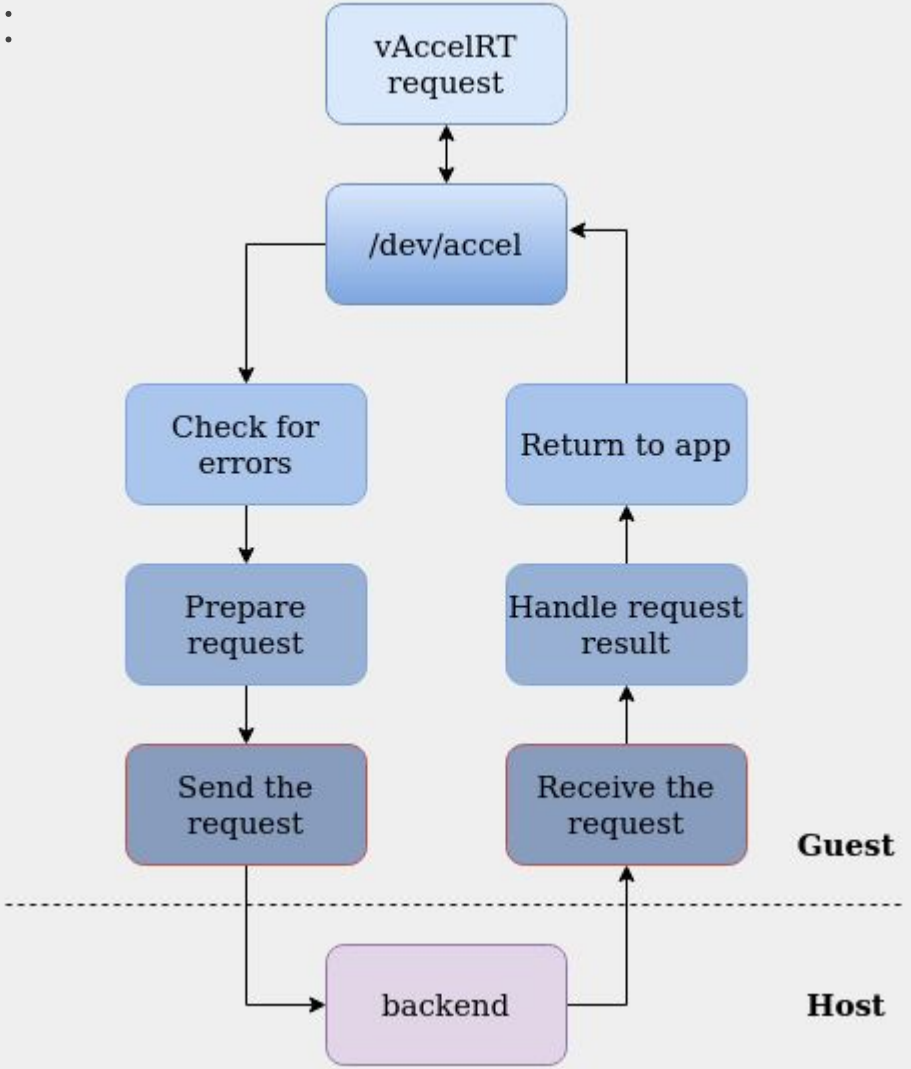
nubificus

- vAccel runtime system
- vAccel virtio front end driver
- Currently supported in 2 unikernels:
  - Unikraft
  - Rumprun
  - More to come (Hopefully...)

# vAccel: front end driver implementation

nubificus

- We can think the driver as it consists of 3 parts:
  a. Character device
  b. Virtio frontend
  c. Glue code
- Character device: used from vAccelRT
  - Open, Ioctl, Close
- Glue code:
  - Prepare the request
- Virtio frontend:
  - Attaches the virtio device
  - Sends the requests
  - Receives the requests



# vAccel: implementation on unikernels

nubificus

- Unikraft port:
  - Easy to register new character device
  - Easy to register new virtio device
  - Unikraft provides similar virtqueue abstractions with linux
- Rumprun port:
  - Easy to register new character device
  - Easy to register new virtio device
  - Dma maps for virtqueue segments

# vAccel: How to use it

nubificus

- We need to prepare 3 components:
  - vAccel runtime on the host
  - QEMU with vAccel backend
  - The unikernel
- Small image classification app, or a REST API app (WIP)
- Unikraft (QEMU-KVM):
  - Easy to use it , from menuconfig
- Rumprun (QEMU-KVM):
  - The config used to bake the application has to include -lrumpdev_virtio_vaccel

# vAccel in FOSDEM’21

**ML inference acceleration on k8s using kata containers & AWS Firecracker**

Where: Containers devroom

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

**ML inference acceleration for lightweight VMMs**

Where: Virtualization & IaaS devroom

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

# 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