Hardware acceleration for Unikernels

A status update of vAccel

Anastassios Nanos, Charalampos Mainas

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

Unikernels are promising

- Fast boot times
- Low memory footprint
- Increased security

Use cases for Unikernels

- Traditional applications
- NFV
- Microservices / Serverless
- ML/AI (?)

Use cases for Unikernels

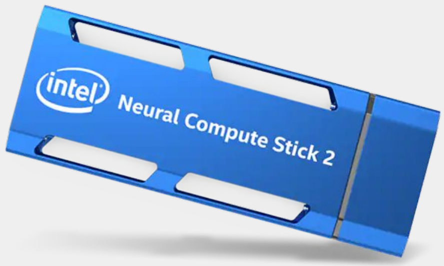
- Traditional applications
- NFV
- Microservices / Serverless
- **ML/AI (?)**

ML/AI workloads

- Heavy frameworks
- Compute-intensive workloads

Hardware acceleration in the cloud and the edge

- Traditional hardware accelerators
  - GPUs
  - FPGAs
- New and specialized Processing Units
  - TPUs
  - ASICs



ML/AI workloads in Unikernels

No support for ML/AI frameowkrs

ML/AI workloads in Unikernels

No support for ML/AI framewokrs

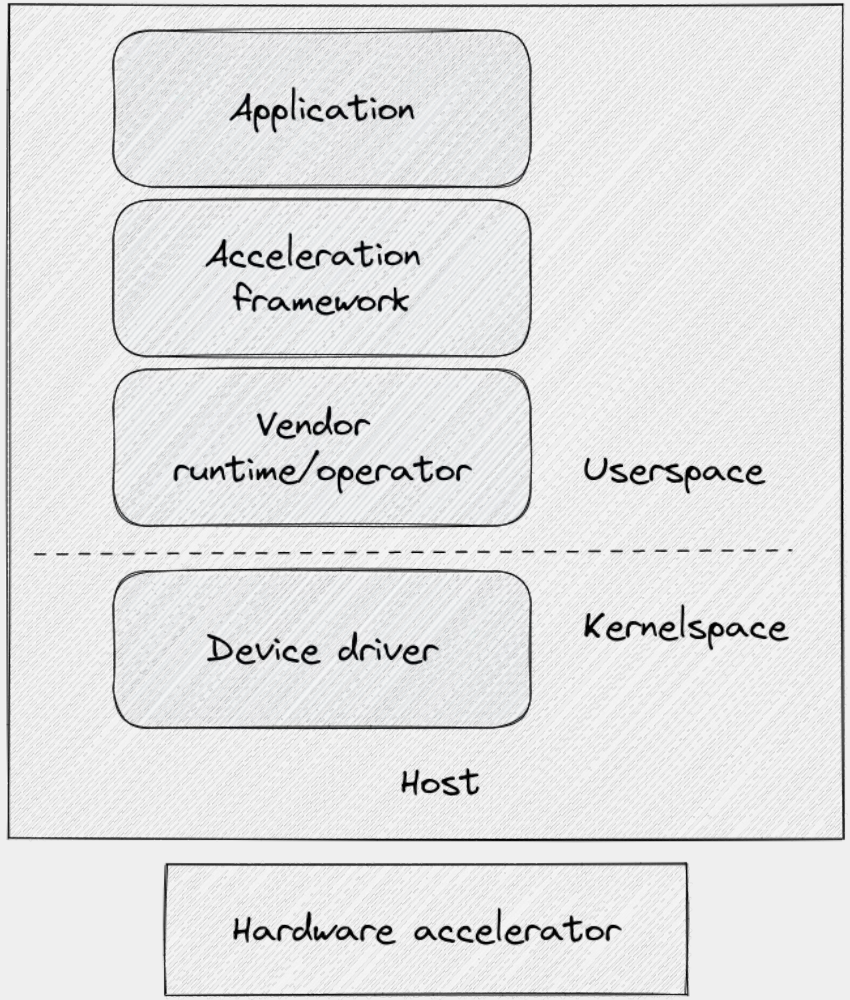
No support for hardware acceleration

Overview

- ~~Motivation~~
- **Virtualization of the hardware acceleration stack**
- Our approach: vAccel
- Insights of vAccel
- Extending vAccel
- Demo

Hardware acceleration software stack

- Acceleration framework
  - OpenCL, CUDA
  - Pytorch, Tensorflow
- Vendor Runtime/Operator
  - Xilinx Runtime, Nvidia GPU operator
- Device driver
  - FPGA, GPU

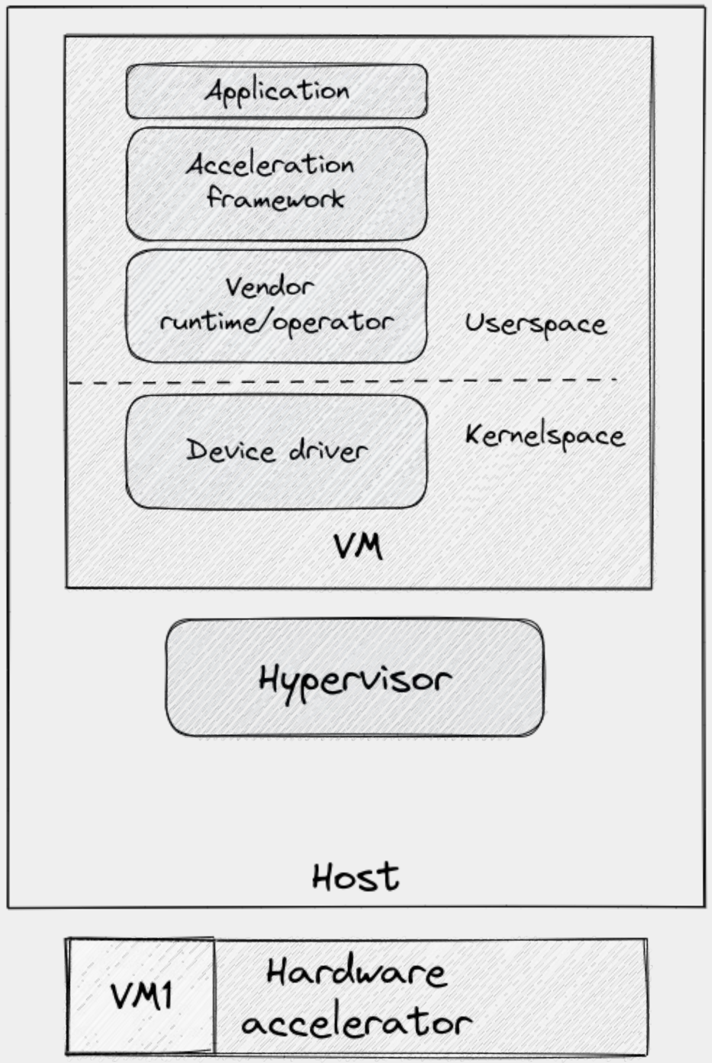


Virtualization of hardware accelerators

- Unikernels are virtual machines
  - Same techniques for device virtualization as in usual VMs
- Device virtualization of hardware accelerators
  - Hardware partitioning
  - Paravirtualization
  - Remote API

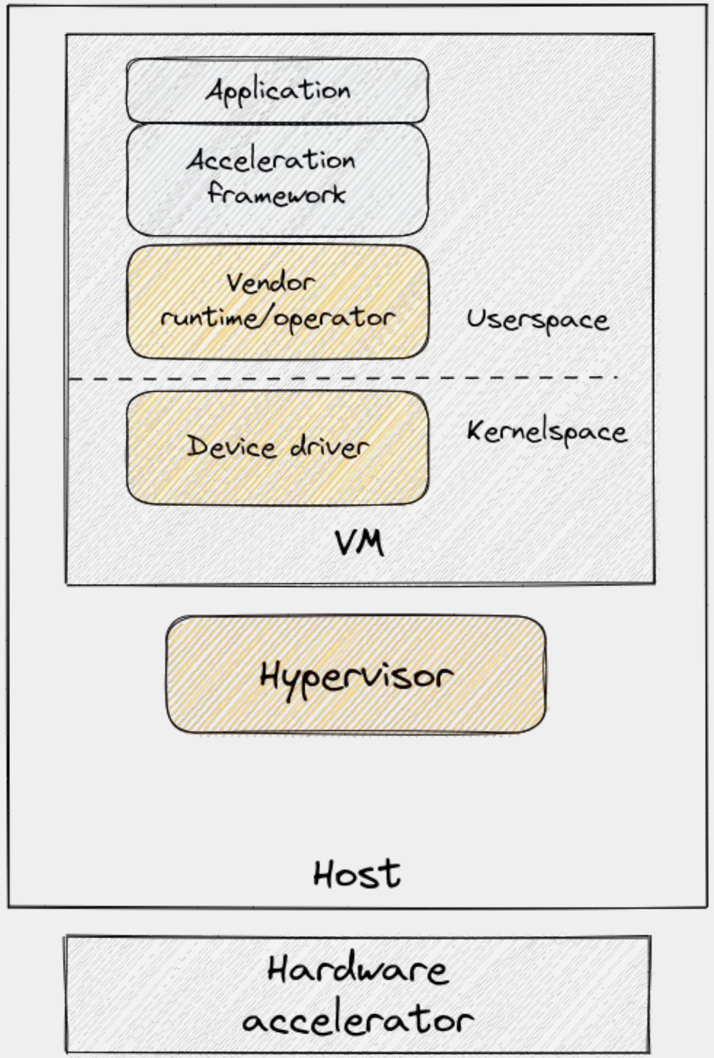
Hardware partitioning

- Split accelerator in partitions
  - Assign a partition to a VM
- Characteristics
  - Entire hardware acceleration stack needs to be in VM
  - Bound to device support/#partitions



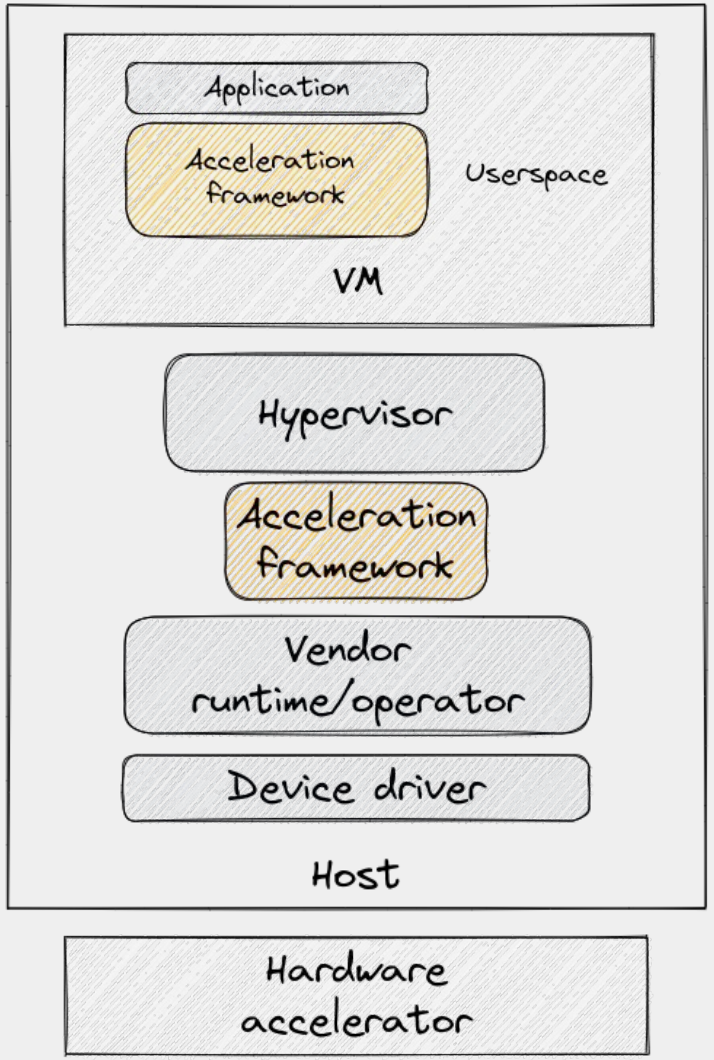
Paravirtualization

- Hypervisor manages the device
    - VMs access device through hypervisor
- Characteristics
    - Significant portion of hardware acceleration stack needs to be in VM
    - Device-agnostic driver, support from hypervisor



Remote API

- Hypervisor manages the device
  - intercept and forward calls to the host
- Characteristics
  - Performance overhead
  - Framework specific



Which one is suitable for Unikernels?

- Hardware partitioning
  - Port of each device driver and rest acceleration stack
- Paravirtualization
  - Port of one device and rest acceleration stack
- Remote API
  - Port only the framework

Porting hardware acceleration frameworks

- Challenges
  - Huge code base
  - Dynamic linking
  - Many dependencies

PyTorch

OpenCL™

NVIDIA.

CUDA

Porting hardware acceleration frameworks

- Challenges
  - Huge code base
  - Dynamic linking
  - Many dependencies

Such frameworks are not suitable for a Unikernel design

Problem statement

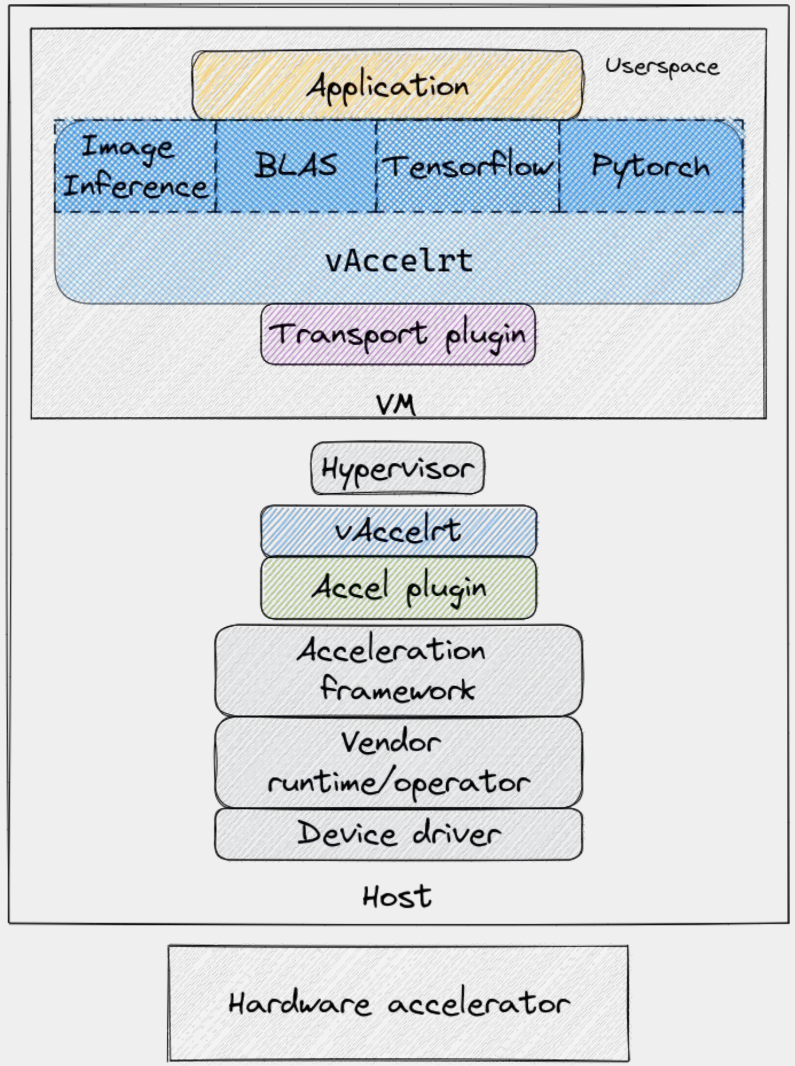
Provide a hardware acceleration solution suitable for Unikernels

Overview

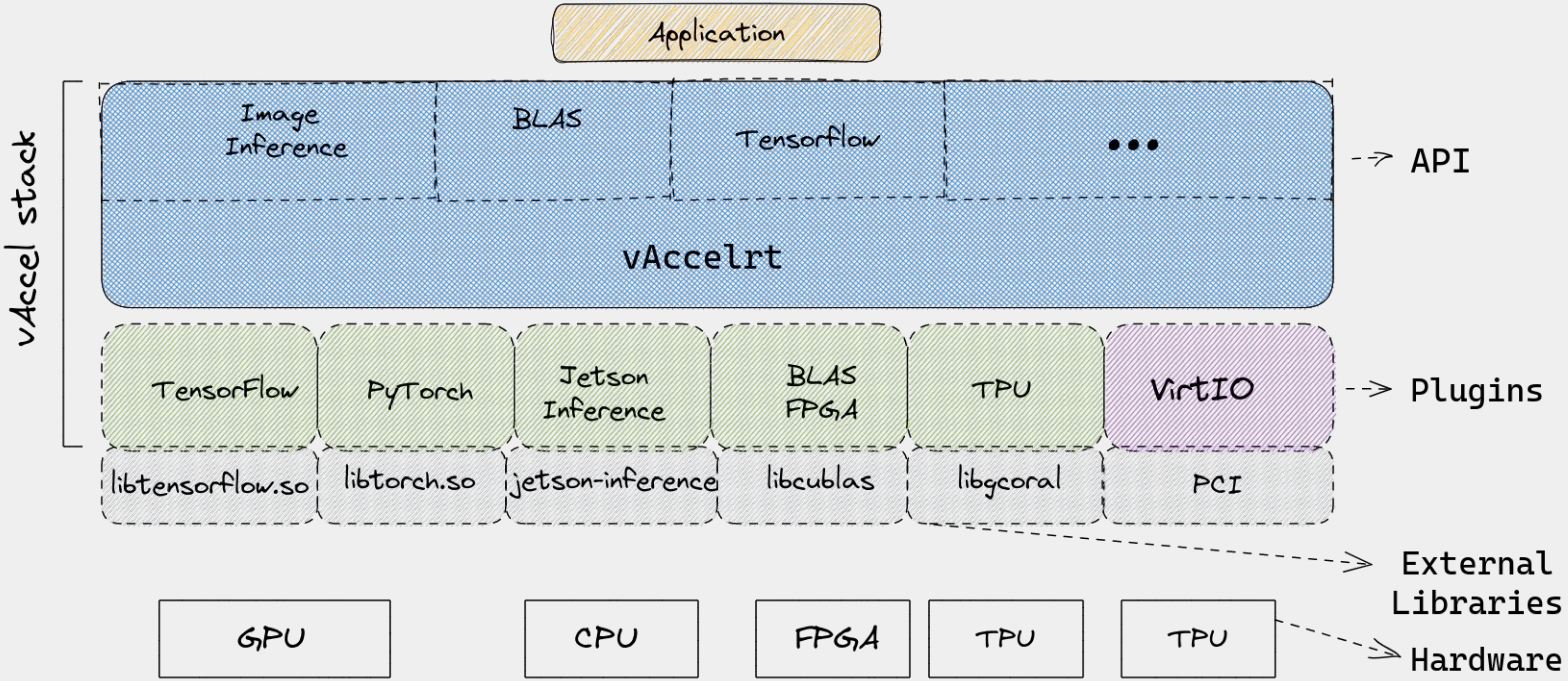
- ~~Motivation~~
- ~~Hardware acceleration stack and virtualization~~
- **Our approach: vAccel**
- Insights of vAccel
- Extending vAccel
- Demo

Our approach: vAccel

- vAccel decouples the function call from its hardware-specific implementation
- Features:
  - Hardware-agnostic API
  - Acceleration in function granularity
  - Portability and interoperability



vAccel overview

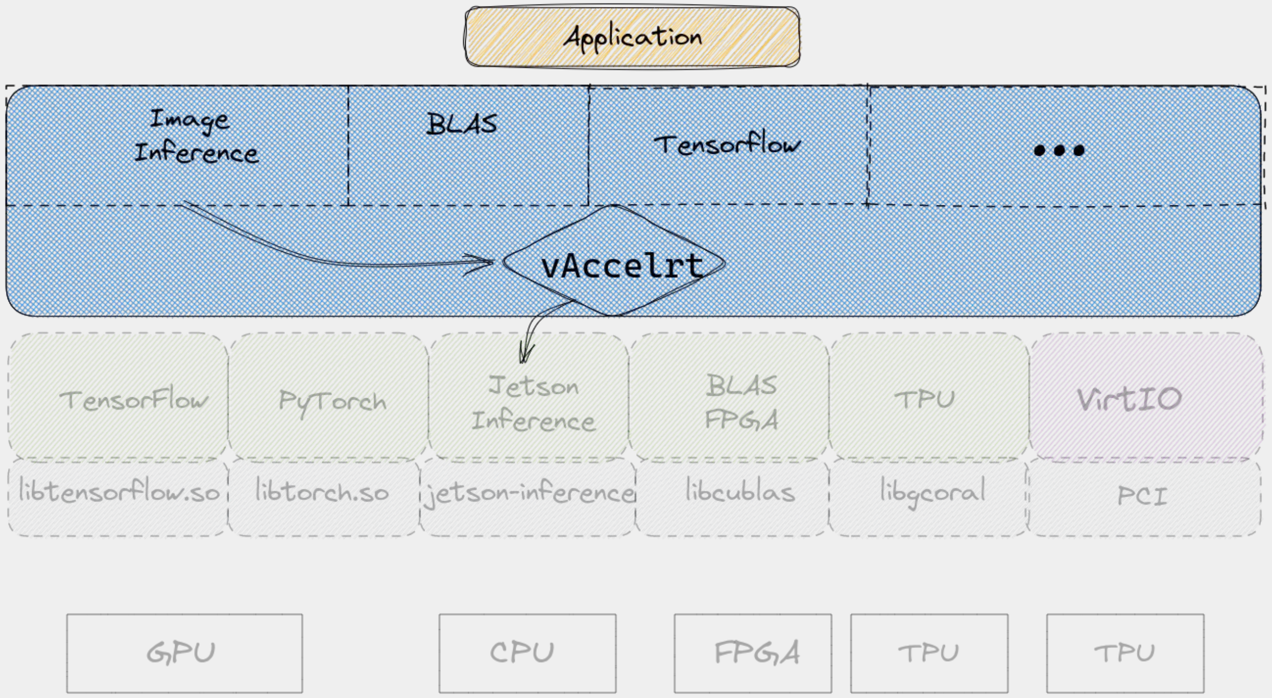


Overview

- ~~Motivation~~
- ~~Hardware acceleration stack and virtualization~~
- ~~Our approach: vAccel~~
- **Insights of vAccel**
- Extending vAccel
- Demo

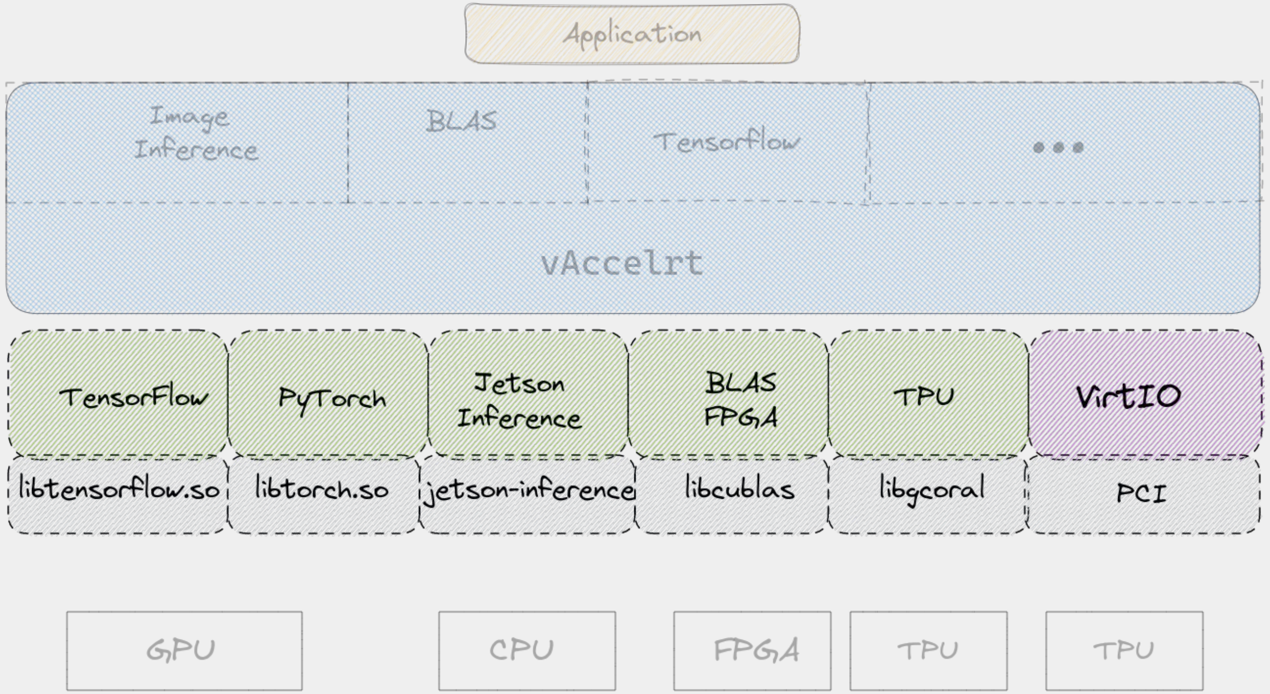
vAccel core runtime library

- Exposes vAccel API
  - Native / Static API, supports accelerated functions (BLAS, classification etc.)
  - Framework bindings (tensorflow, pytorch)
- Receives and manages requests
  - Forwards requests to available implementations



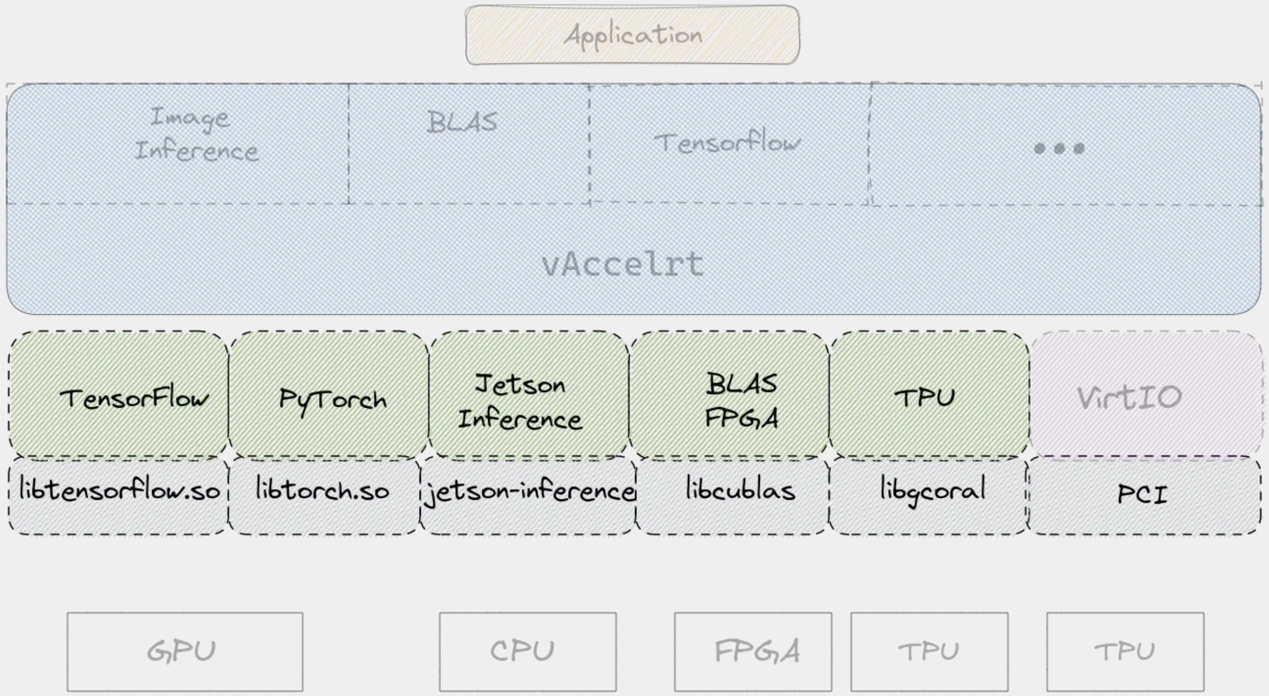
vAccel plugins

- Glue code between vAccel and actual implementation / framework
  - Plugins for acceleration frameworks
  - Plugins for transport layers



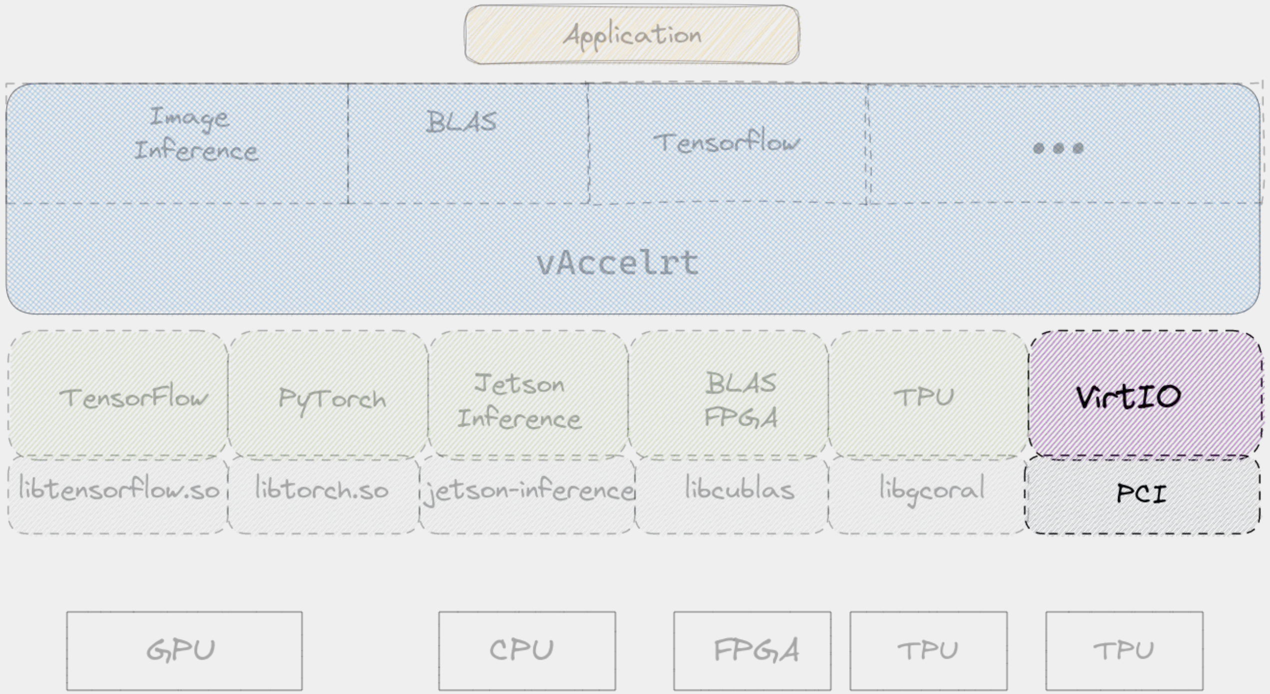
vAccel plugins for acceleration frameworks

- Glue code between vAccel and function implementation
- Acceleration frameworks (Tensorflow, Pytorch)
- Hardware / framework implementation of an operation (Image classification, BLAS etc.)



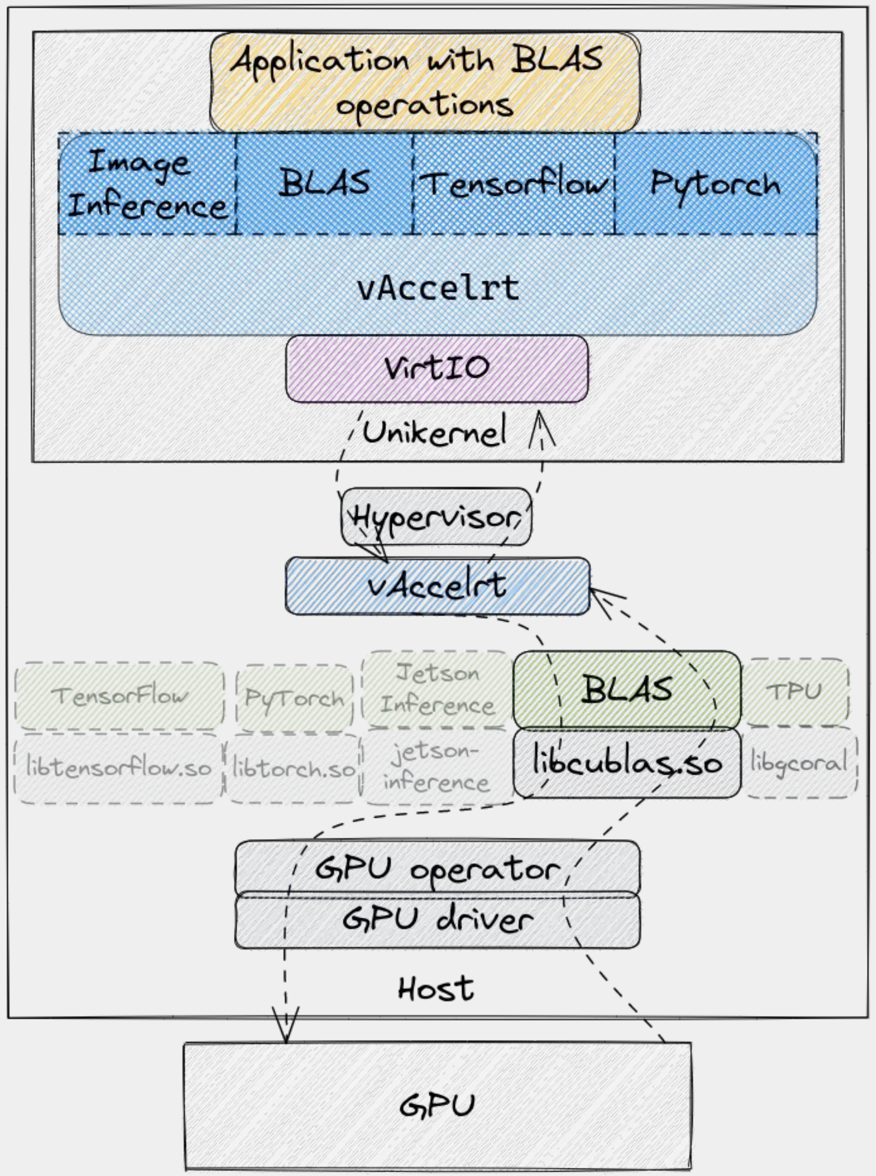
vAccel plugins for transport layers

- Forward requests from guest to host
- Transport layers:
  - VirtIO (PCI / MMIO)
  - Socket interface:
    - AF_VSOCK (virtio-vsock)
    - AF_INET (TCP sockets)



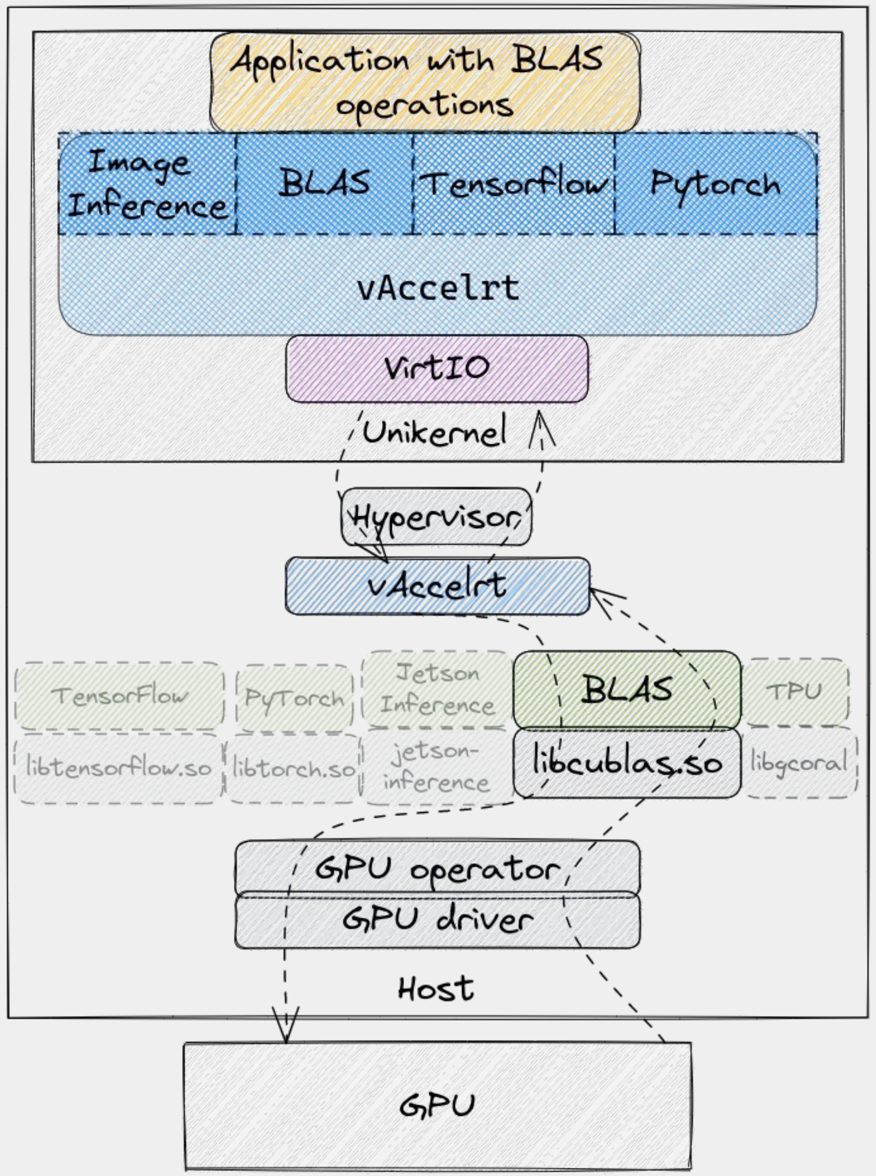
vAccel in Unikernels

- Ideal abstraction and easy to port
  - Specialization of hardware acceleration
  - Thin layer of C code without any dependencies
  - Only transport plugins are needed



vAccel in Unikernels

- Portability / Interoperability
  - Identical native/unikernel code
  - Access to various frameworks and hardware without code changes

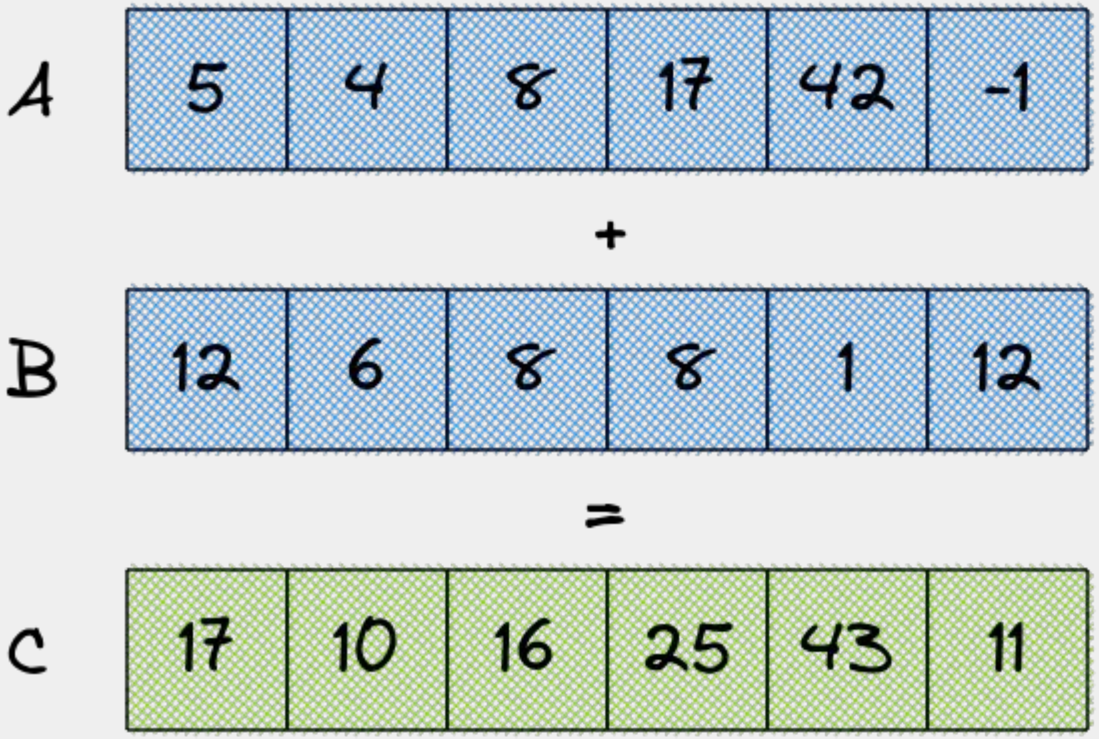


Overview

- ~~Motivation~~
- ~~Hardware acceleration stack and virtualization~~
- ~~Our approach: vAccel~~
- ~~Insights of vAccel~~
- **Extending vAccel**
- Demo

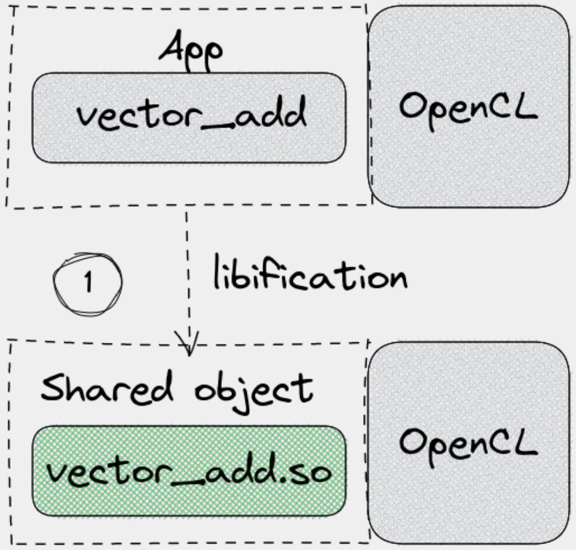
Porting applications in vAccel

- Example: vector addition in OpenCL
    - Setup vector_add bitstream in FPGA
    - Transfer arrays A and B in FPGA
    - Invoke FPGA kernel
    - Transfer array C from FPGA



Porting applications in vAccel

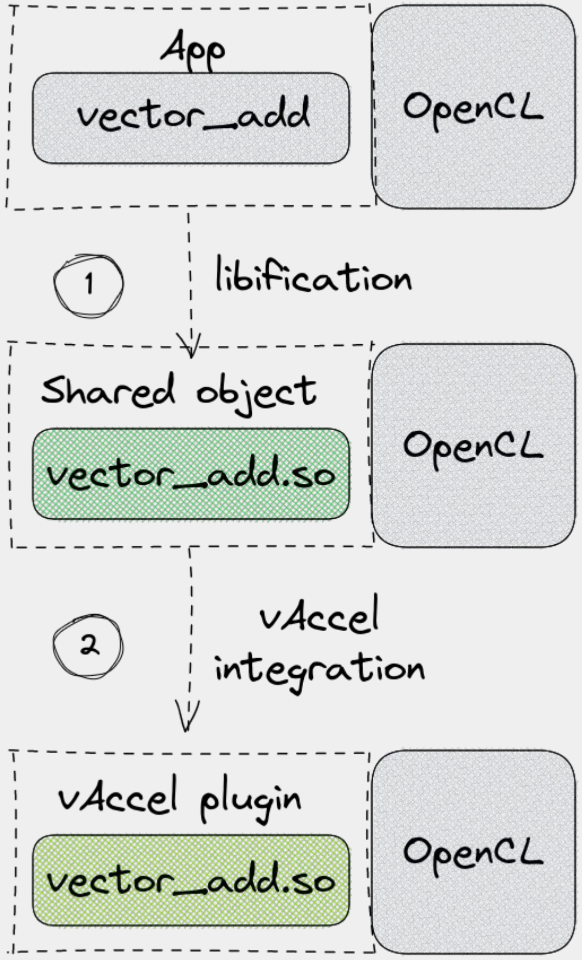
1. "Libification" of the application



Porting applications in vAccel

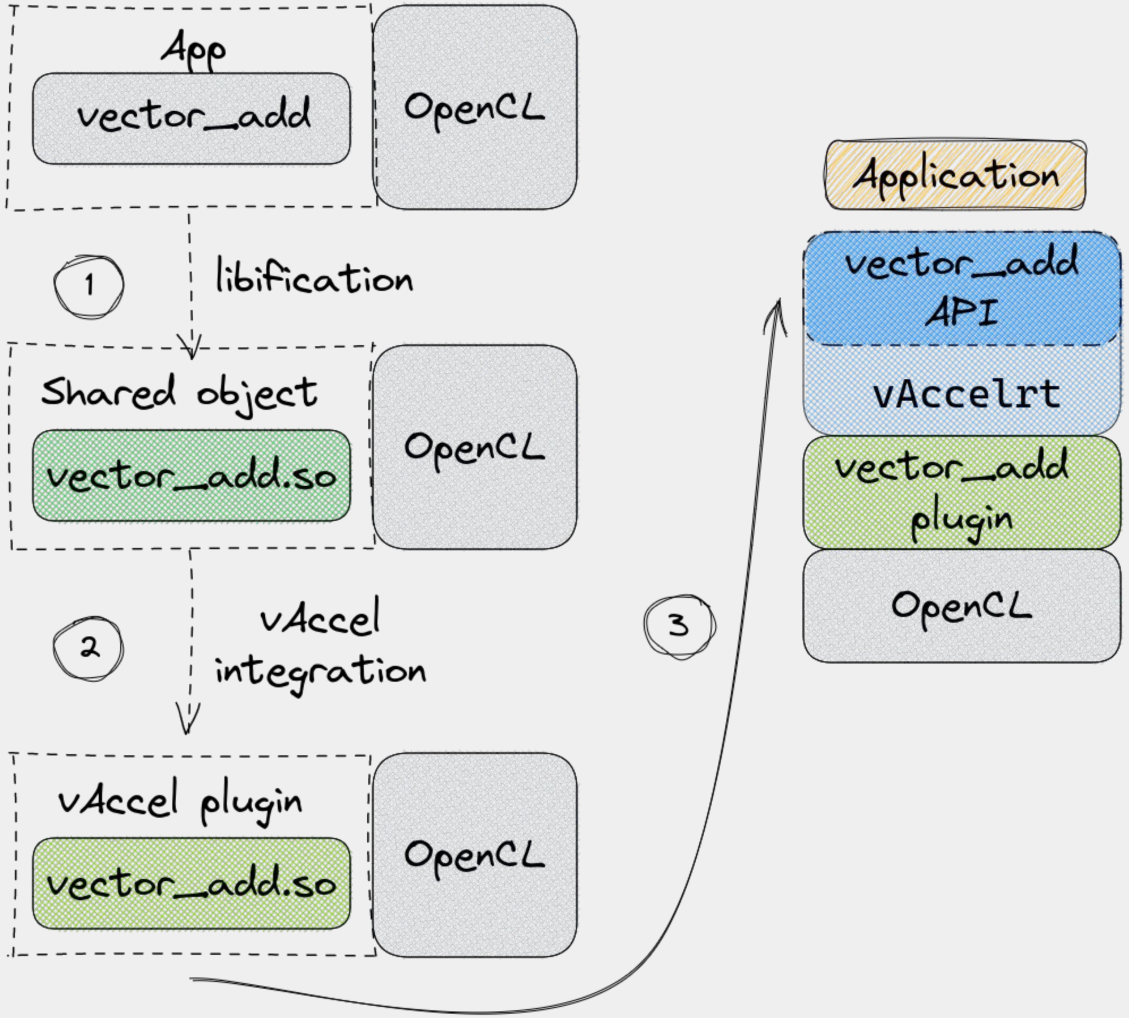
1. "Libification" of the application

2. Integrate the library in Vaccel as a plugin



Porting applications in vAccel

1. “Libification” of the application
2. Integrate the library in Vaccel as a plugin
3. Expose the new function in vAccelrt



Overview

- ~~Motivation~~
- ~~Hardware acceleration stack and virtualization~~
- ~~Problem statement~~
- ~~Our approach: vAccel~~
- ~~Insights of vAccel~~
- **Demo**

Demo

- Usage of acceleration frameworks from Unikraft
  - BLAS cuda (CPU/GPU)
  - Jetson-inference (GPU)
  - OpenCL (FPGA)

Current state of vAccelrt

- vAccelRT v0.5.0 released!
  - https://github.com/cloudkernels/vaccel/releases/tag/v0.5.0
- Language Bindings for C/C++, Python, Rust, TF
  - https://docs.vaccel.org/language_bindings/
- Simple plugin API
  - https://docs.vaccel.org/plugin

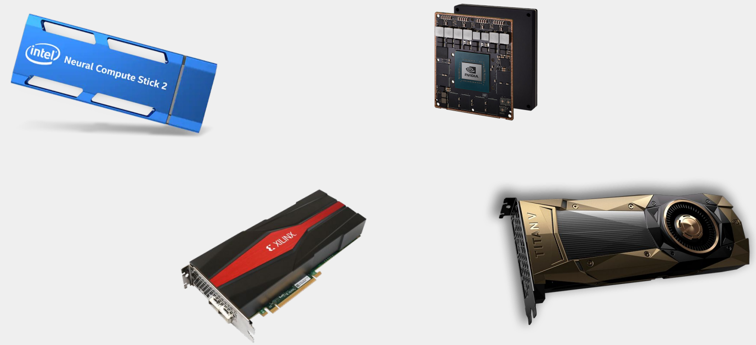
vAccel systems support

- Hypervisors:
  - QEMU (VirtIO & vsock)
  - Rust-VMM clones:
    - AWS Firecracker (VirtIO & vsock)
    - Cloud Hypervisor (vsock)
    - Dragonball (vsock)

- Unikernels:
  - Unikraft
  - Rumprun
- Integration with k8s, kata-containers and OpenFaaS

vAccel support for frameworks & hardware

- Acceleration frameworks:
  - Jetson-inference (dusty-nv)
  - Tensorflow / Pytorch
  - TensorRT / OpenVINO
  - OpenCL / CUDA
- Hardware:
  - GPUs (NVIDIA RTX/T4, NVIDIA Volta/Maxwell etc.)
  - Edge TPUs/NPUs (MyriadX, Coral, AMlogic etc)
  - FPGAs UC50/200, PYNQ



Summary

- Hardware acceleration stacks are huge and complicated
- vAccel abstracts the heterogeneity of the hardware and the frameworks
- Perfect fit for Unikernels

Summary

- Hardware acceleration stacks are huge and complicated
- vAccel abstracts the heterogeneity of the hardware and the frameworks
- Perfect fit for Unikernels

Try it out!!!

https://vaccel.org & https://docs.vaccel.org

https://github.com/cloudkernels/vaccelrt

https://github.com/nubificus/vaccel-tutorials

This work is partly funded by the European Union's Horizon 2020 research and innovation programme under Grant Agreements no 871900 (5G-COMPLETE) & 101017168 (SERRANO)

https://unikraft.org/community/hackathons/2023-03-athens/
https://unikraft.org

Register now: https://forms.gle/a315sJrzRQV8rZdz8

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