October 10-12, 2024

Technical overview & performance evaluation of Virtio-loopback

Timos Ampelikiotis - Alvise Rigo - Daniel Raho

Virtual Open Systems

VITAMIN-V

This work has received funding from the European Union's Horizon Europe research and innovation programme under Grant Agreement No. 101093062 (VITAMIN-V).

E-mail: contact@virtualopensystems.com

Presentation index

- Virtio-loopback scope
- Objectives
- Overview
- Basic components
- Existing HAL technologies
- Benchmarks – evaluation
- Conclusion & Future work

Virtio-loopback scope

- Automotive
- IoT
- Cloud

What virtio-loopback was developed to solve

Existing problem:

- HW manufactures need to protect their patents and proprietary solutions.
- For not exposing the internals of their solutions they need to provide their drivers as a part of a kernels image binary
- That leads them to maintain and port their drivers for many versions of the Linux kernel → Maintainability costs

New approaches:

- Avoiding the extra maintainability cost new approaches for creating user-space drivers are used instead (easy porting between kernel)
- The need also for a well defined Hardware Abstraction Layer (HAL) is required for serving those drives and standardize the way that are communicating with the applications.

Virtio-loopback objectives

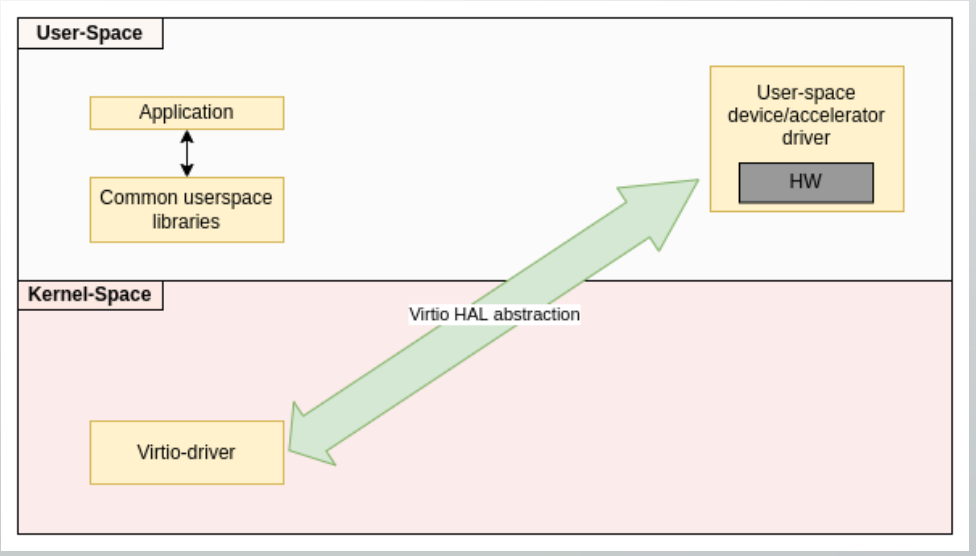
Virtio-loopback was design to assists the needs of the industry and HW manufactures to:

- Hide the internal of their HW solutions
- Increase the portability of their drivers
- Decrease the maintainability costs
- Standardize the ABI between drivers and application

Virtio-loopback overview

In an effort to address industry's requirements Virtio-loopback was developed by Automotive Grade Linux (AGL) and Virtual Open Systems as a HAL between application and user-space drivers based on the following principle:

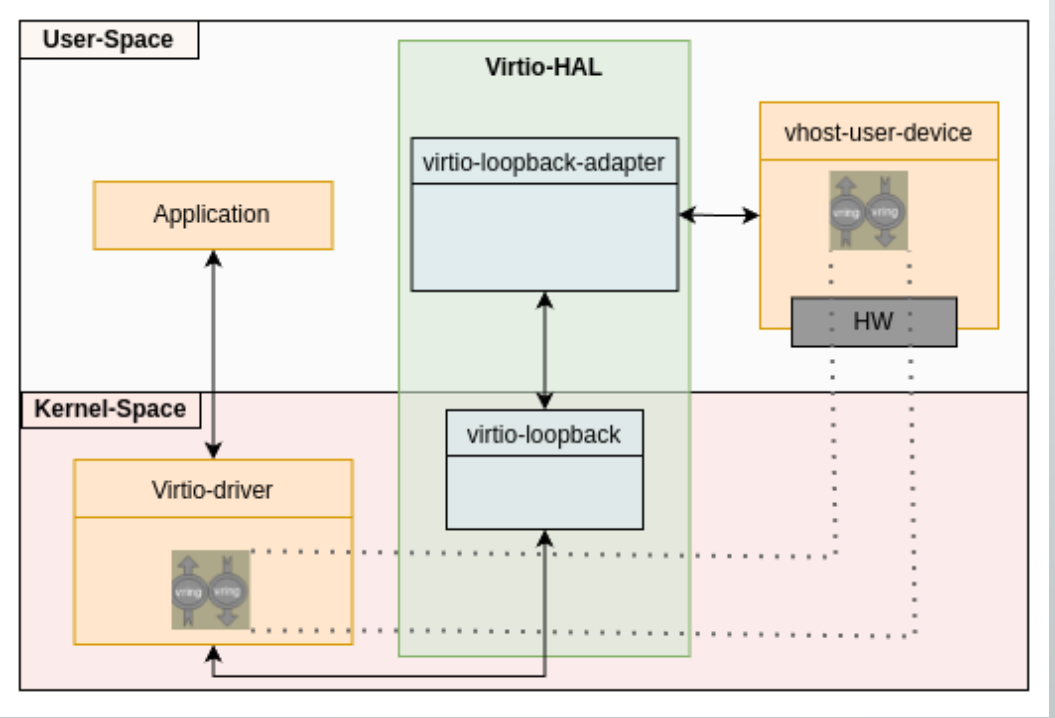
- Based in existing open standard: Virtio, vhost-user
  - Does not introduce any modification
- Fully transparent between applications and drivers
  - Application using those standards are fully portable to this technology
- It does introduce any significant performance overhead
  - Zero copy principle
- Architecture independent
  - Works for x86, Arm, Risc-v



Virtio-loopback: Basic components

The virtio-loopback design consists of:

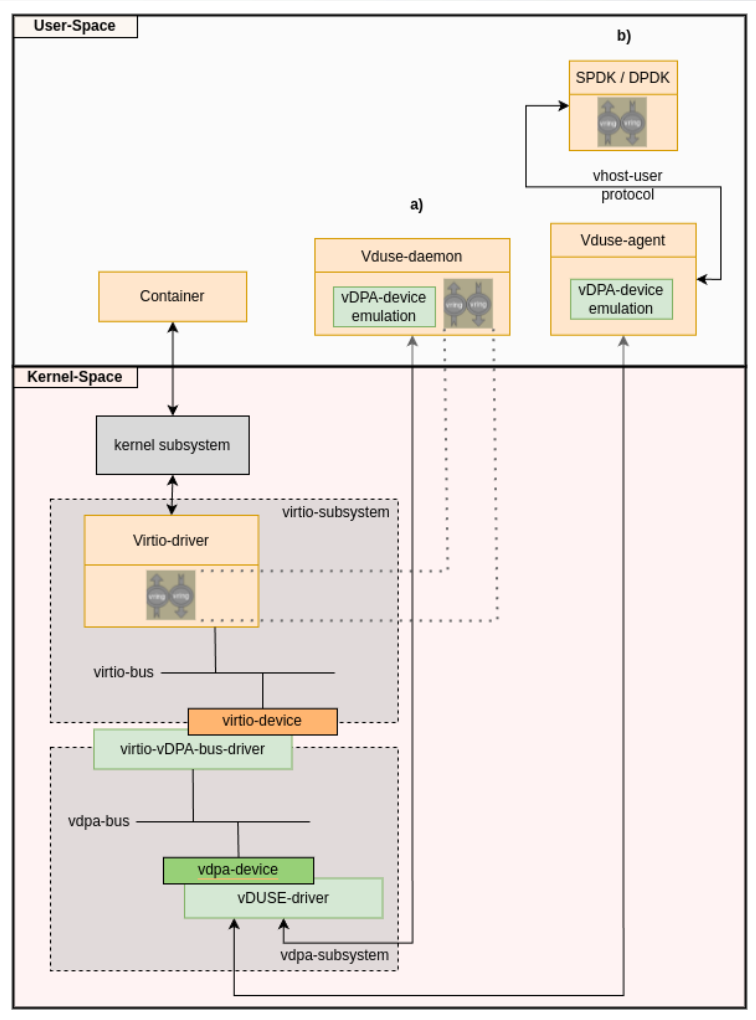
- A kernel module
  - Acts both as virtio-transport and management driver
  - Reroutes the virtio messages to the user-space
  - Makes sure the user-space components can access the virtio drivers data to the kernel
- A user-space application
  - Bridging the communication with the user-space (vhost-user) driver
  - Delivers notification between the
  - module and the vhost-user component.



Existing technologies: vDUSE

vDUSE (vDPA device in userspace) is part of a greater kernel framework called vDPA. Was design to work for both container and VM workloads and targets to move the emulation of the virtio-device into the user-space and bridge its communication with the virtio-drivers into the host kernel. It consists of:

- A kernel module which based on the vDPA framework is able to reroute the virtio traffic into the user-space
- A user-space component which facilitates the virtio-device emulation

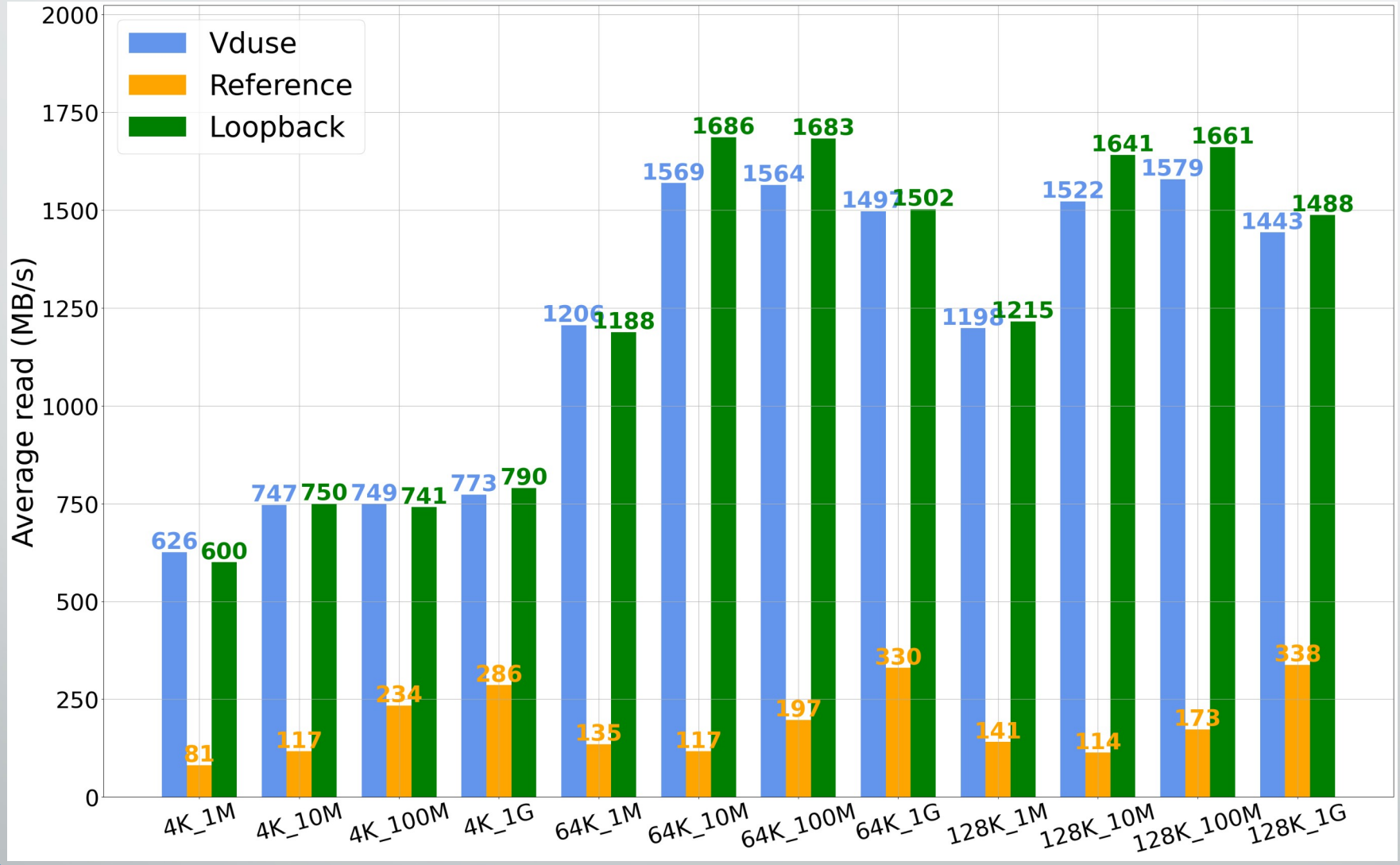


Virtio-loopback performance evaluation - setup (1)

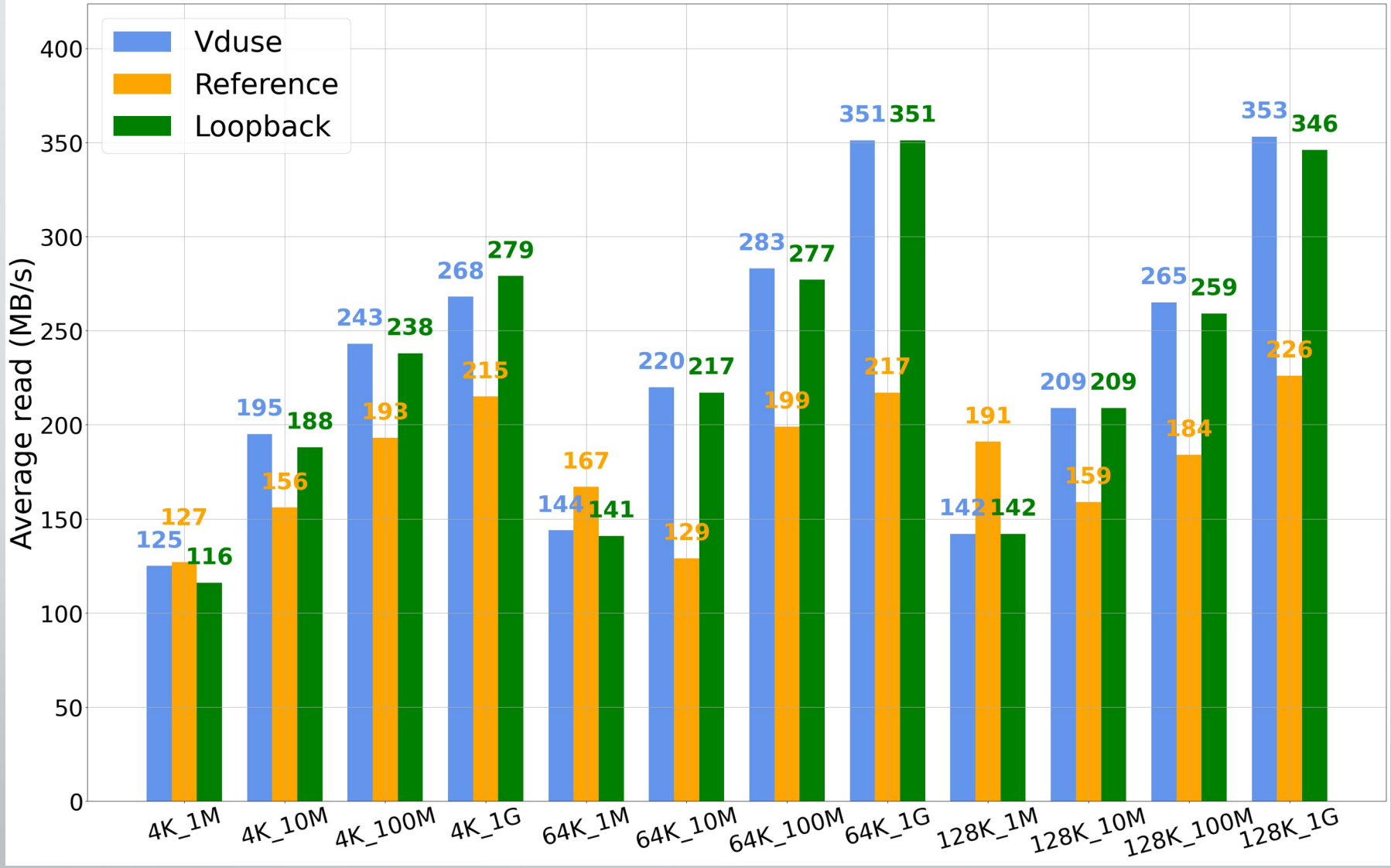
The benchmarks were perform:

- In three different architecture devices (x86, arm65, risc-v64)
- Utilizing the "fio" tool
- Focusing on virtio-blk (block device) case since it expose best the latency and throughput characteristics of both designs

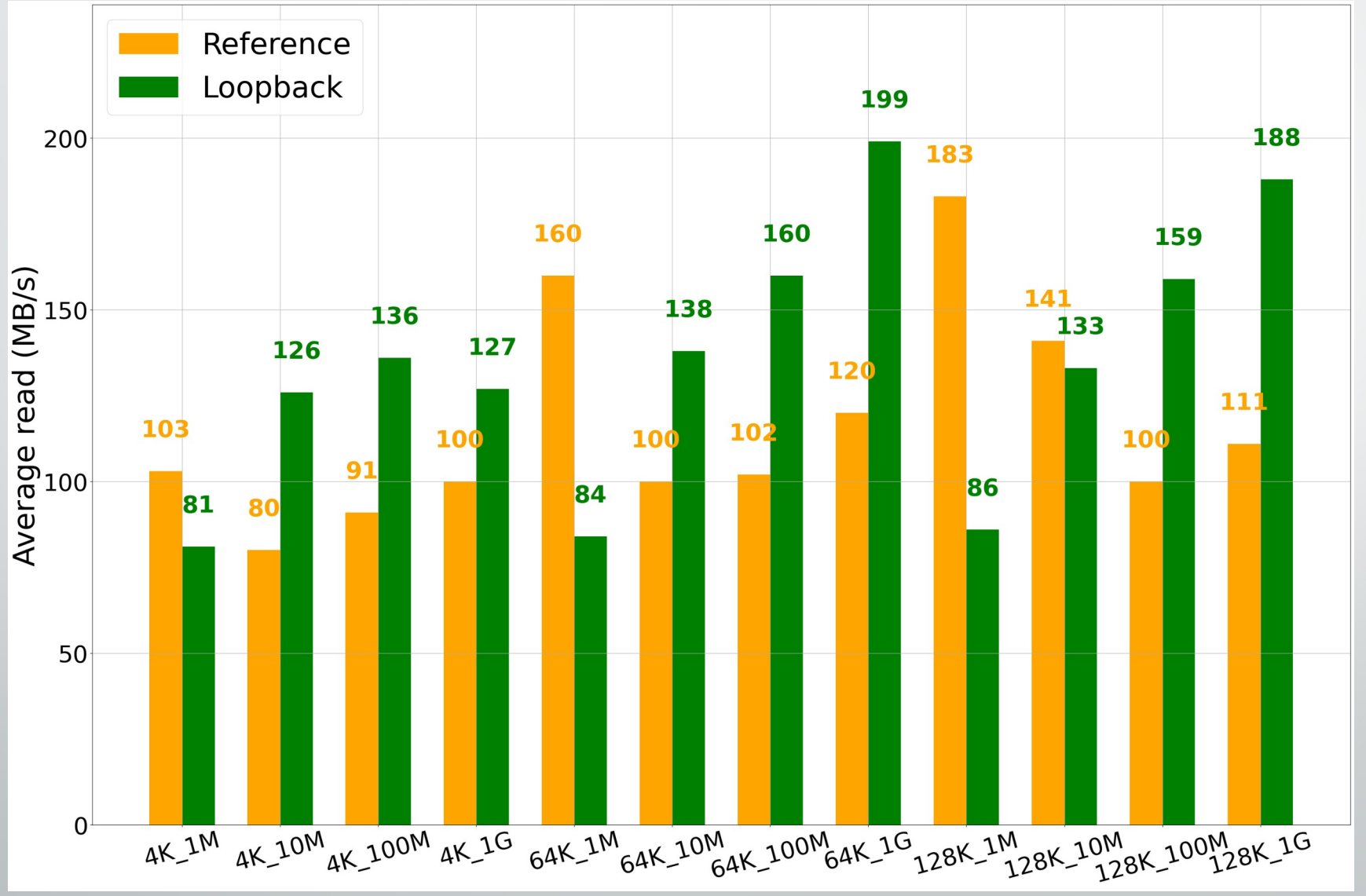
Virtio-loopback performance evaluation - x86 (2)



Virtio-loopback performance evaluation - Arm64 (3)



Virtio-loopback performance evaluation - Risc-V64 (4)



Virtio-loopback performance evaluation – conclusion (5)

Based on the results:

- The two solutions performs similarly in all three* cases
  - Disclosure: vDUSE was not run for risc-v64 case due to the driver missing for the platform supported kernel (vDUSE was introduced in v5.15 and the kernel used for that test was v5.10)
- The differences observed are mainly due to the different nature of the data sharing mechanism of the virtio-device to the user-space
- Interesting observation: The reference case (performing the fio benchmark directly onto the source image file as a reference point measurement) seems to be overtaken by the virtio-based solutions while the size of the read/write operations increases
  - That could be explained as the two solution act as a "caching layer" between the benchmark application and the destination image file.
- Overall, the benchmark results proves that Virtio-loopback architecture is aligned with today's performance standards.
  - It is also clear that certain aspects of virtio-loopback can be further improved when it come to performance on low-power devices (raspberry-pi4, lichee64 etc.).

Virtio-loopback: Conclusion & Future work

Overall, based on the multi-platform benchmark results and the comparison with vDUSE, the performance of the virtio-loopback design seems to be aligned with today.s standards.

Therefore it could stand a viable choice for HW manufactures and application developers to be used as a virtio-HAL layer between application and user-space drivers.

Future work will focus on:

- Updating the notification mechanism → decrease latency
- Evaluating further Virtio-loopback's memory mapping mechanism with "perf tool". Address weaknesses and update the mechanism → increase throughput.
- Supporting more virtio-devices like virtio-net, spi, i2c, vsock etc.
  - current support for virtio-blk, rng, input, gpio, can, console, sound.

THANK YOU FOR YOUR ATTENTION!

In case of questions, please do not hesitate to contact the authors by the following e-mails: contact@virtualopensystems.com