CHEOPS 2023

Next-Gen Cloud Storage: Leveraging DPUs to Virtualize File System Services

Peter-Jan Gootzen 🇨🇭 🇳🇱, Jonas Pfefferle 🇺🇸, Radu Stoica 🇨🇭, Animesh Trivedi 🇳🇱

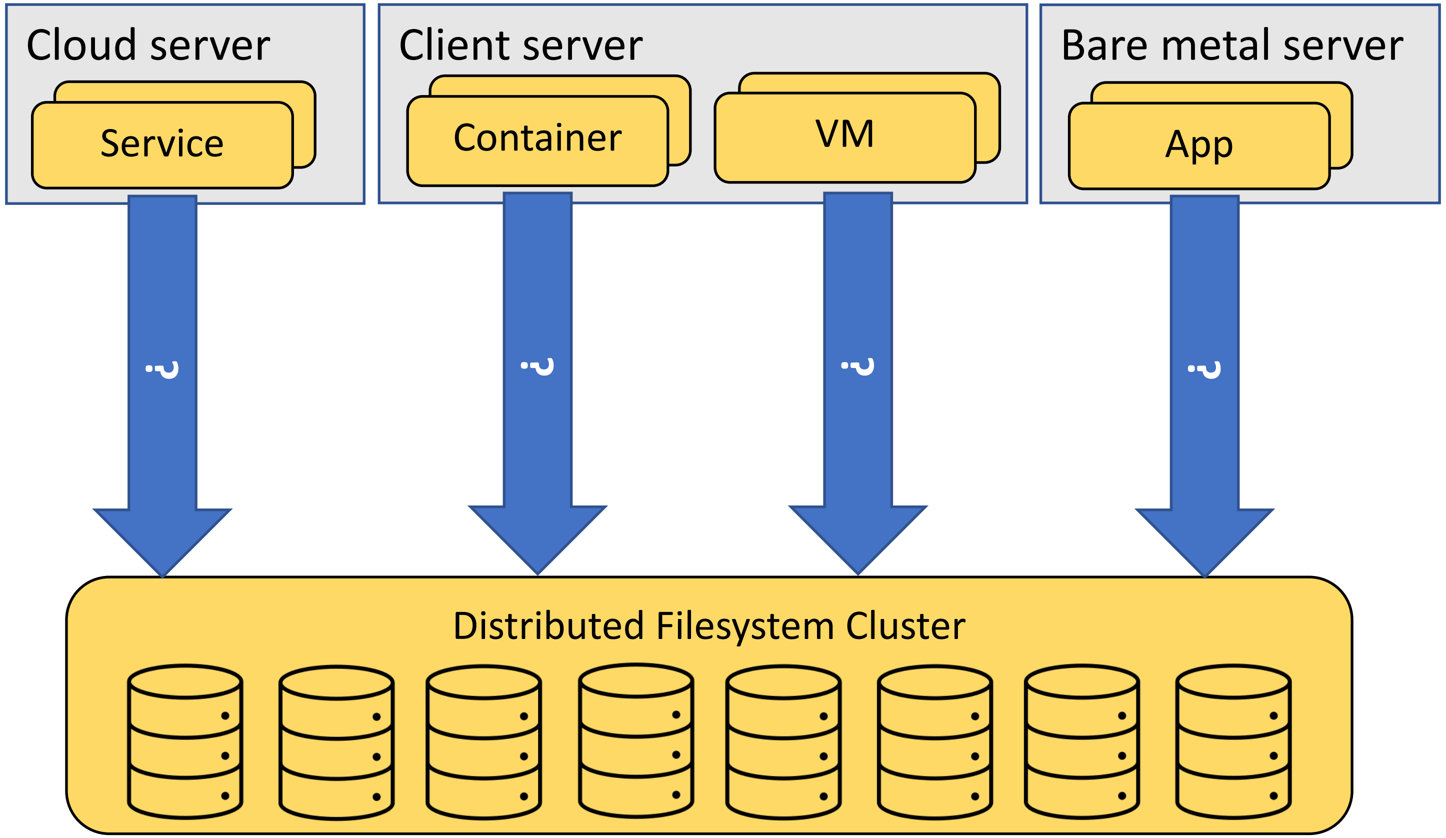
IBM Research
Zurich 🇨🇭 and Yorktown 🇺🇸

IBM Research

Vrije Universiteit Amsterdam 🇳🇱
AtLarge Research

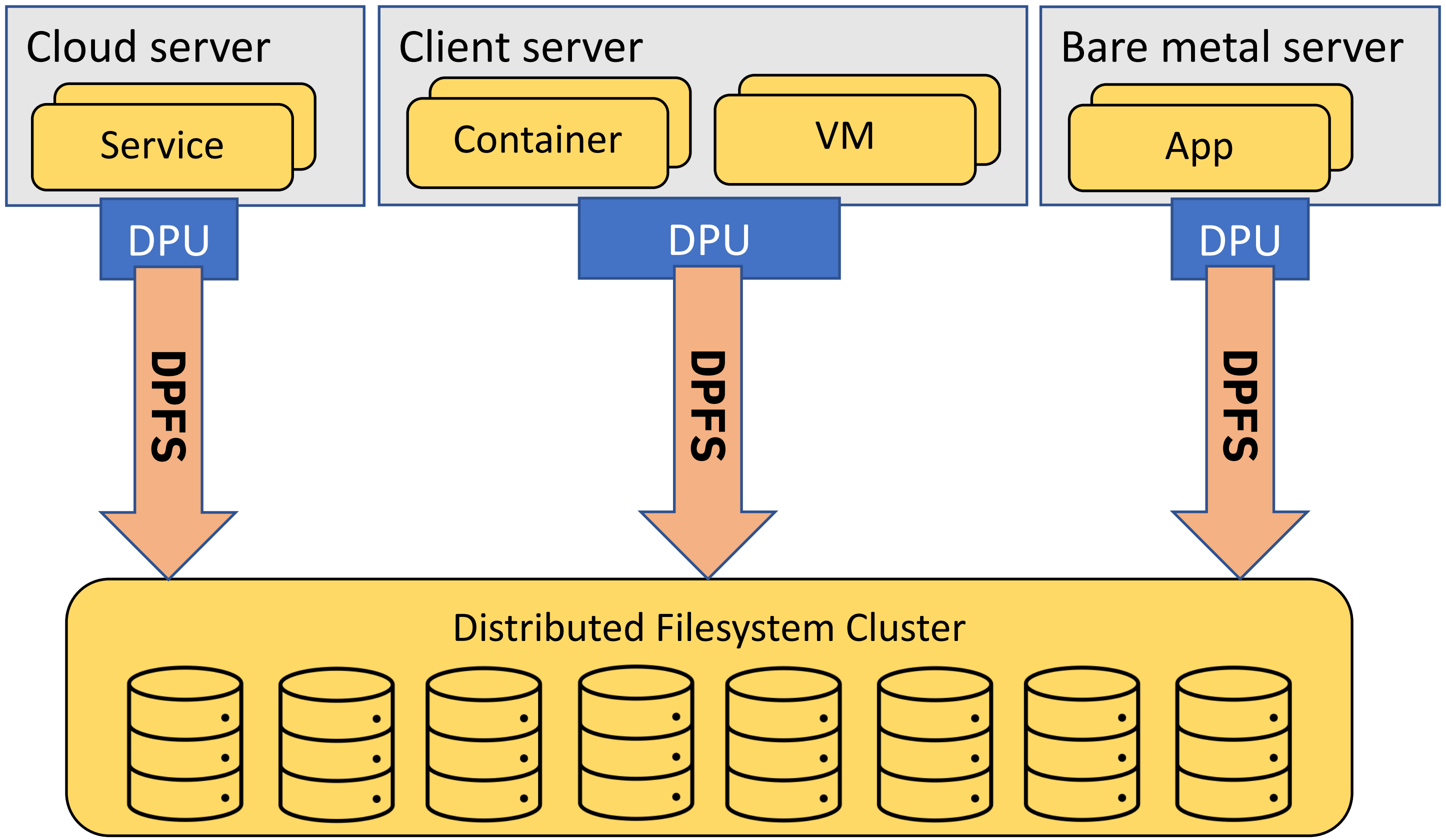
VU VRIJE UNIVERSITEIT AMSTERDAM

How to consume FS services in a Cloud?



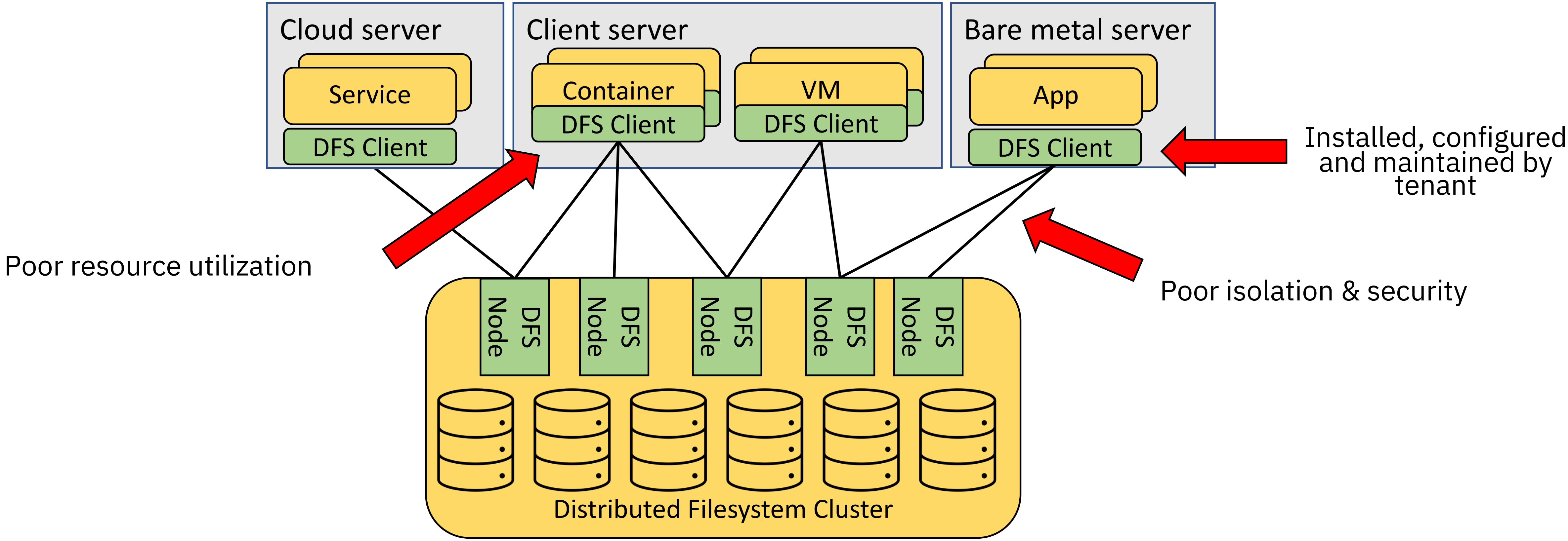
Efficiency			Management			Security	
Performance	Overhead	Multi-tenancy	Support all cloud clients	Client transparency	Operator control	Attack surface	Network isolation

DPU-Powered File System Virtualization



Efficiency			Management			Security	
Performance	Overhead	Multi-tenancy	Support all cloud clients	Client transparency	Operator control	Attack surface	Network isolation

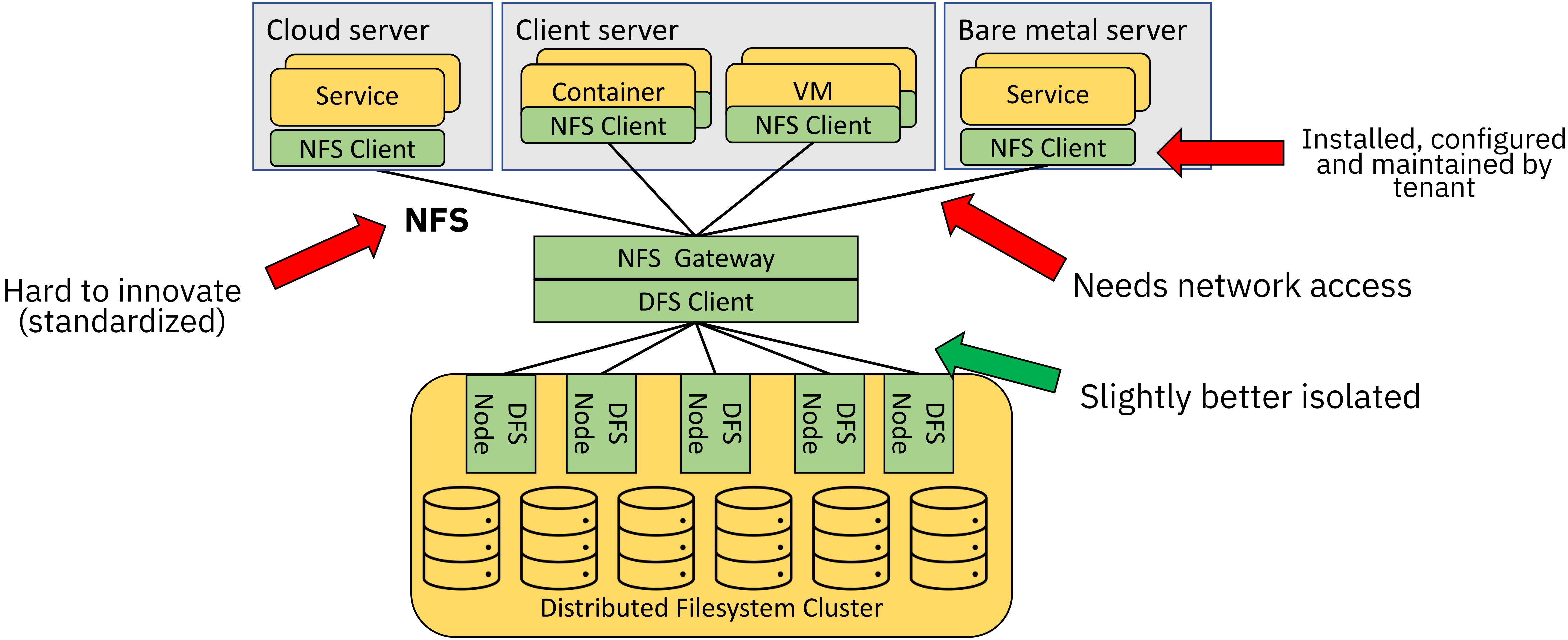
Option 1: *Traditional* Distributed File System client



Examples: Spectrum Scale, CephFS, etc.

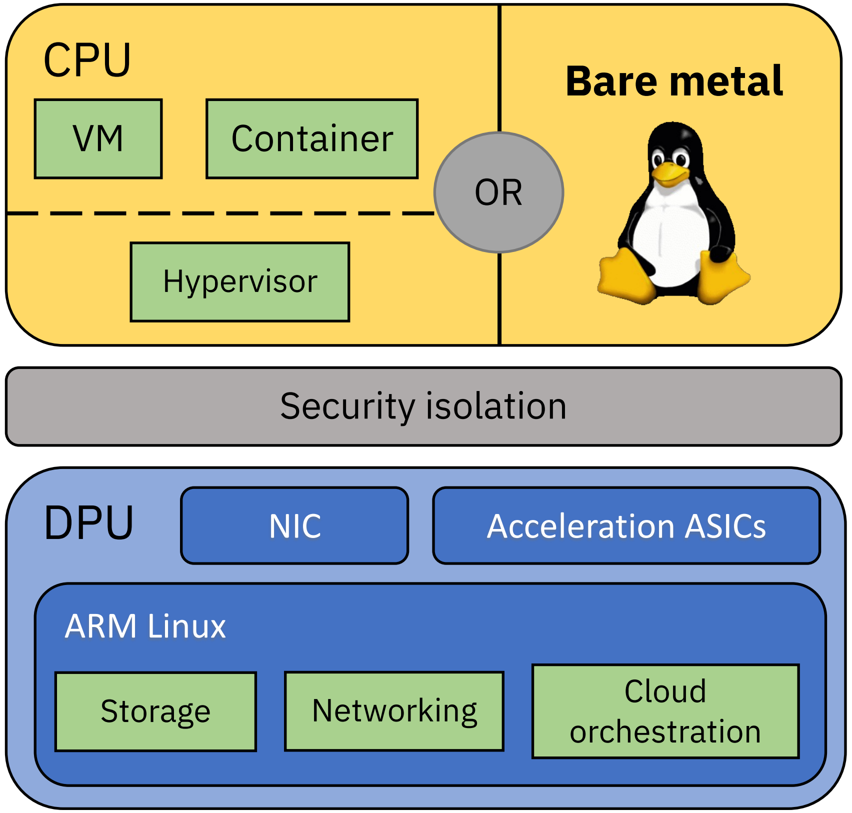
Efficiency	Management	Security

Option 2: NFS gateway for Cloud File Systems



Efficiency	**Management**	**Security**

The DPU-powered Cloud ☁️



- Also known as *SmartNIC* or *Infrastrucure Processing Unit (IPU)*
- "A NIC with compute and offload capabilities baked in"
- We focus on DPUs with a CPU

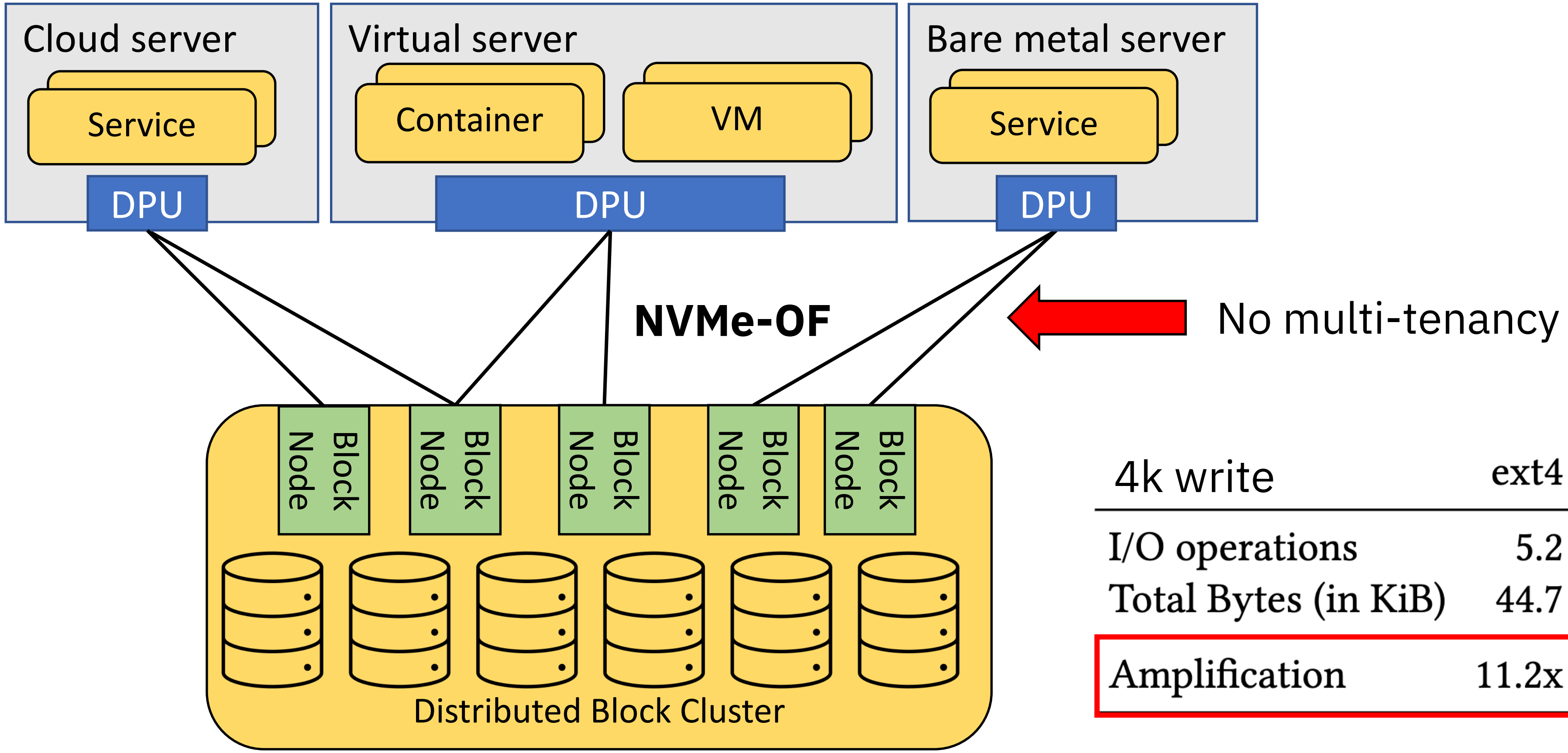
Offloading using DPUs:

✔ Block storage devices (NVMe and virtio-blk)

✔ Networking (virtio-net & programmable switch)

✖ File systems **"DPFS" to fill the gap**

Option 3: Remote Block Storage

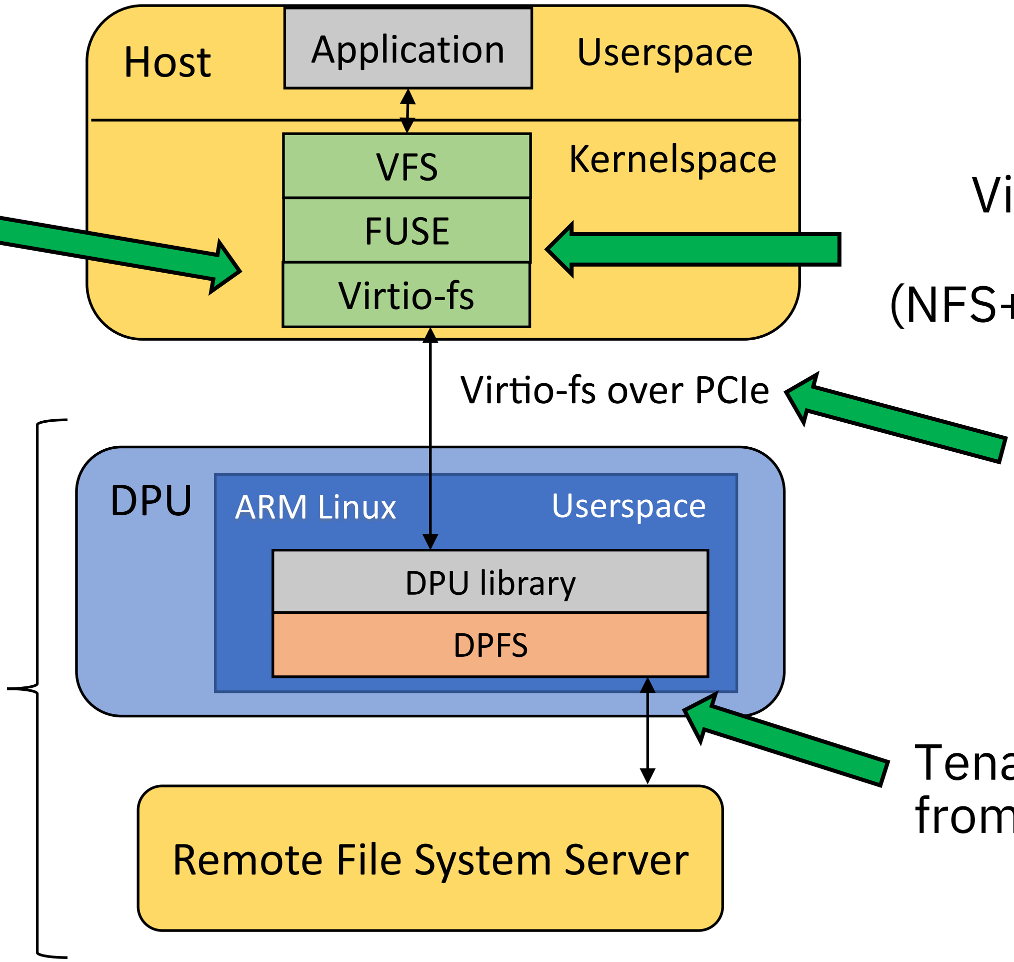


4k write	ext4	ext4 + NVMe-oF	XFS	Btrfs
I/O operations	5.2	13.7	3	4.6
Total Bytes (in KiB)	44.7	46.8	12	125.3
Amplification	11.2x	11.7x	3x	16x

Efficiency | **Management** | **Security**

The high-level FSvirtualization stack

- No configuration
- Works on bare metal
- Transparent consumption of any FS



Virtio-fs = 13k LoC
vs.
(NFS+TCP/IP) = 181k LoC

Multi-tenancy (SR-IOV)

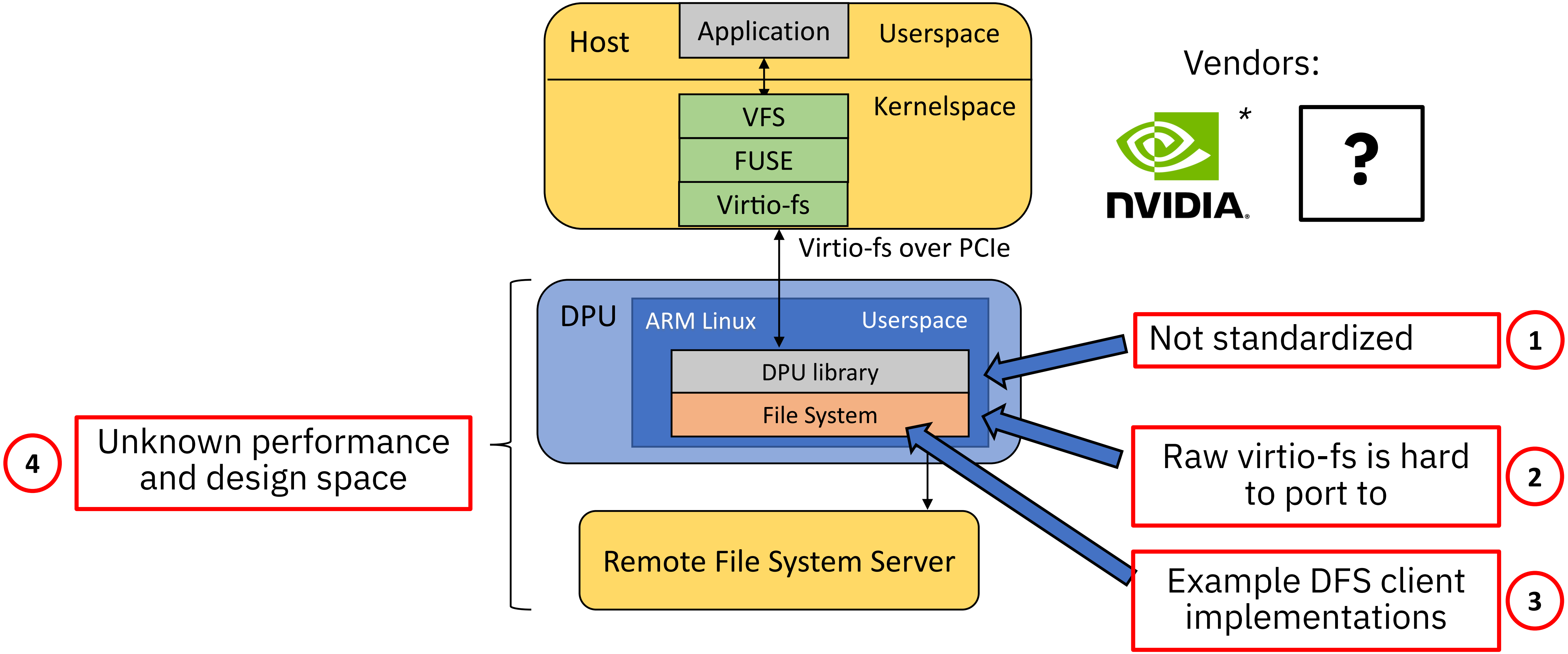
Maximum flexibility and full control for hardware specialization

Tenant completely isolated from FS client and network

Efficiency			Management			Security	
Performance	Overhead	Multi-tenancy	Support all cloud clients	Client transparency	Operator control	Attack surface	Network isolation

IBM

Challenges that **DPFS** solves

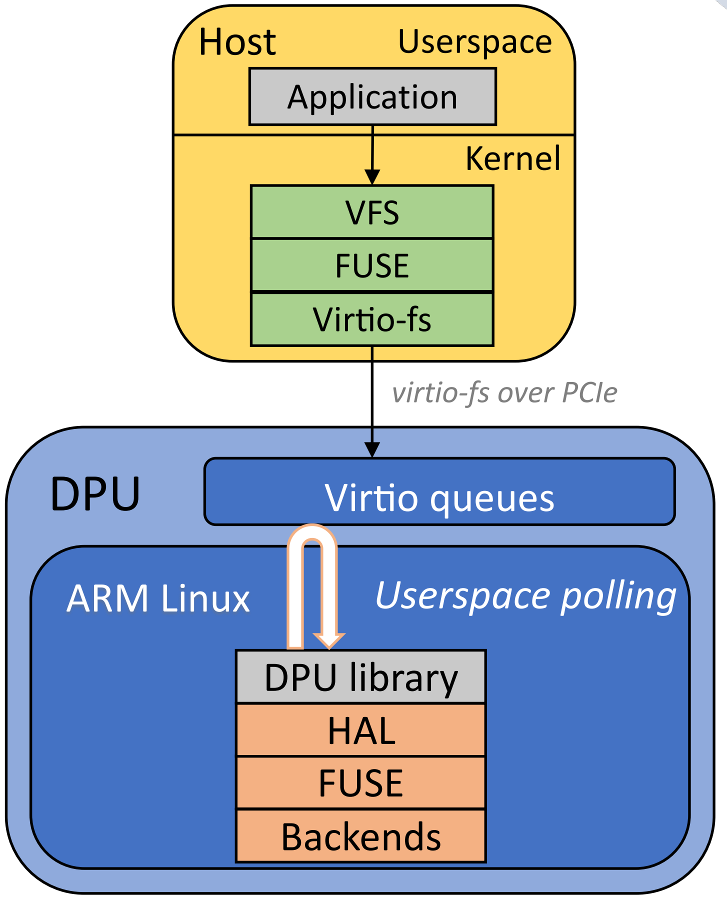


Kick-start open research and adoption!

The **DPFS** framework: **D**PU-**P**owered **F**ile **S**ystems

Architecture:

1. Hardware Abstraction Layer
2. FUSE API
3. Several backends



The **DPFS** framework: **D**PU-**P**owered **F**ile **S**ystems

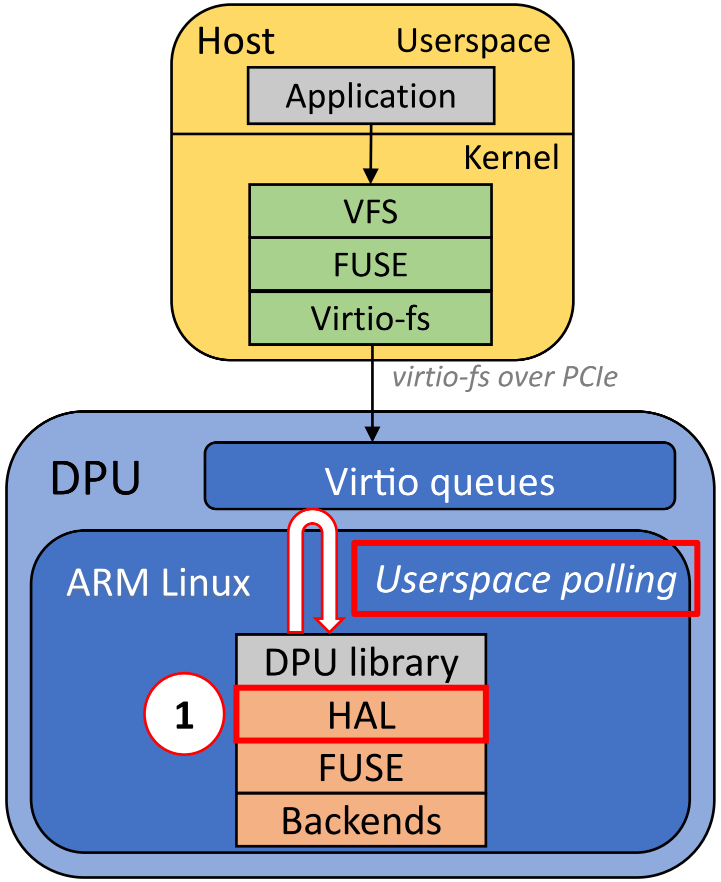
Architecture:

1. **Hardware Abstraction Layer**
2. FUSE API
3. Several backends

Vendors:

*

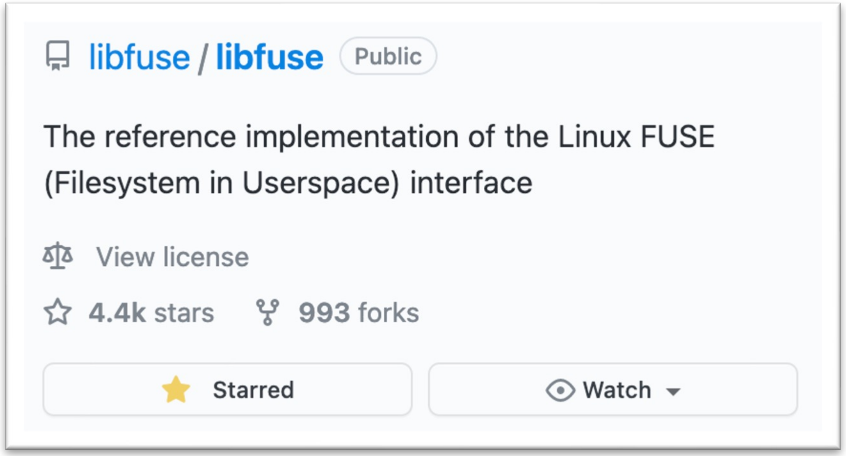
?



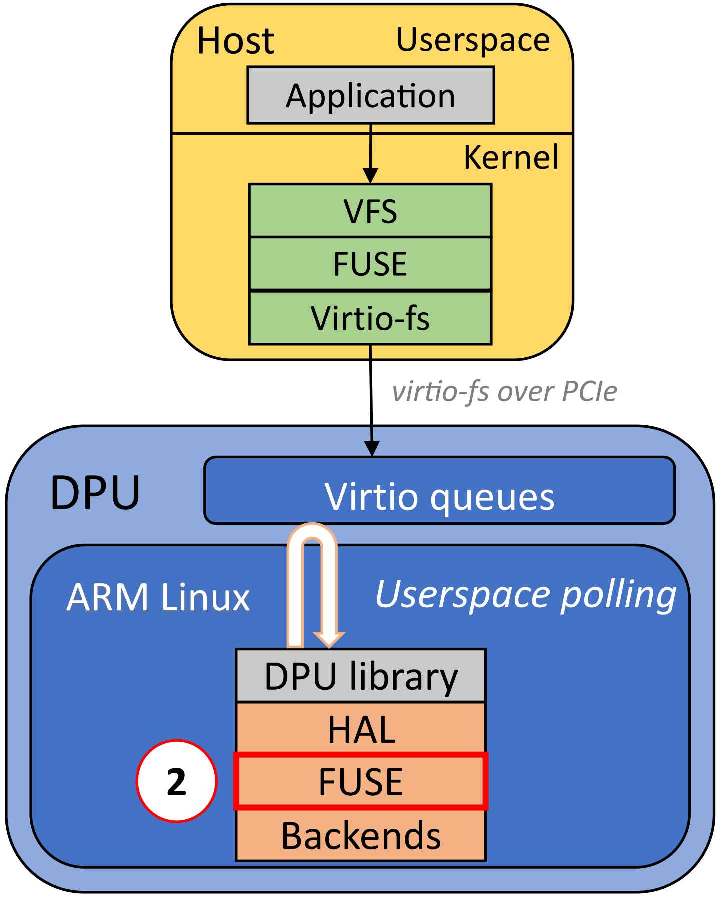
The **DPFS** framework: **D**PU-**P**owered **F**ile **S**ystems

Architecture:

1. Hardware Abstraction Layer
2. **FUSE API**
3. Several backends



API ~equal, but no multithreading *yet*

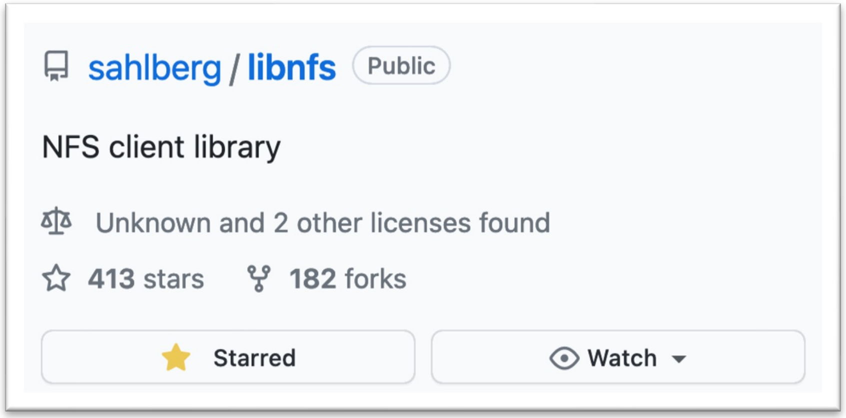


The **DPFS** framework: **D**PU-**P**owered **F**ile **S**ystems

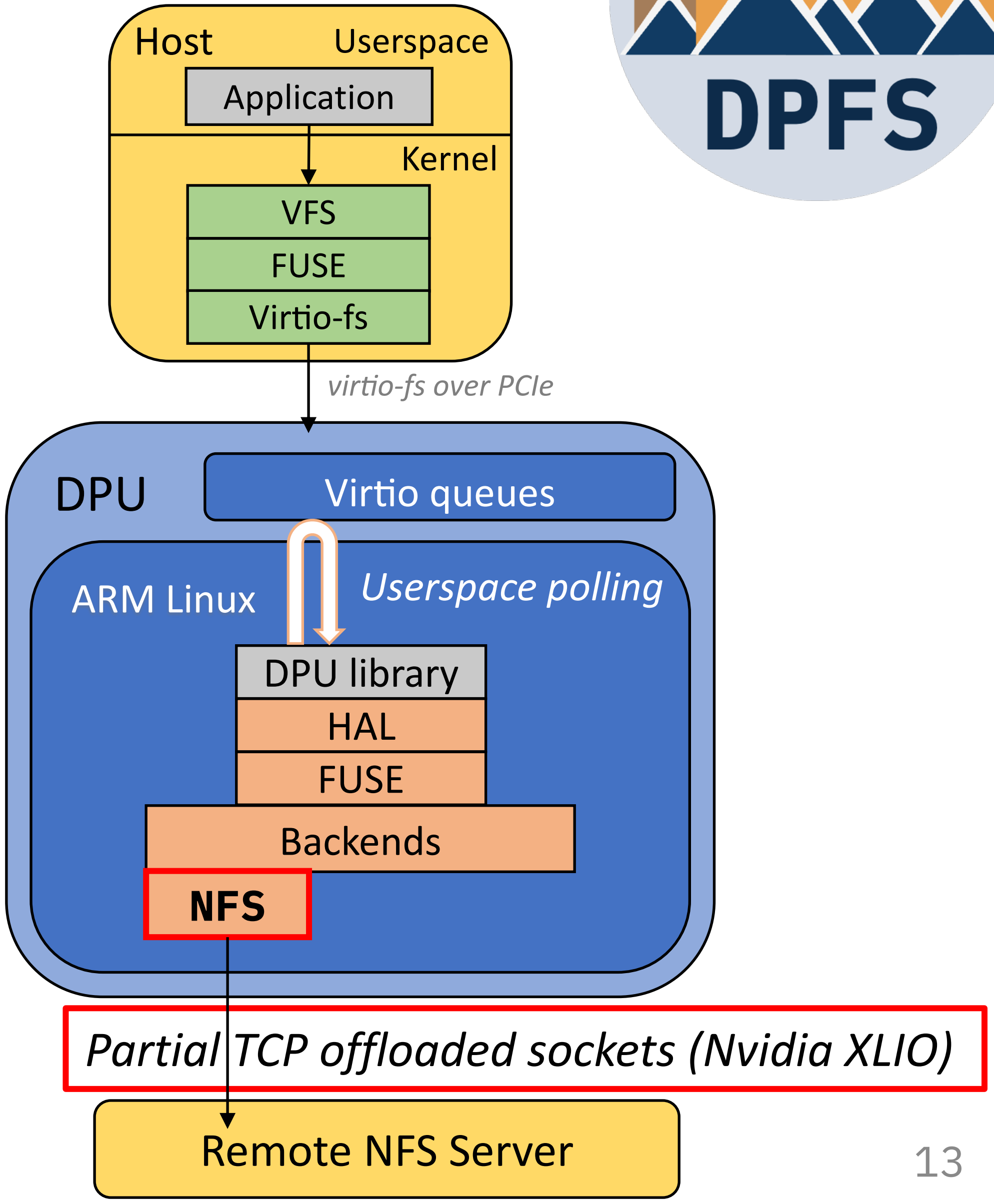
Architecture:

1. Hardware Abstraction Layer
2. FUSE API

3a **Several backends: NFS**



Userspace NFS v4.1



The **DPFS** framework: **D**PU-**P**owered **F**ile **S**ystems

Architecture:

1. Hardware Abstraction Layer
2. FUSE API

3b **Several backends: NFS, KV**

Appears in *SIGOPS Operating Systems Review*, Vol. 43, No. 4, December 2009, pp. 92-105

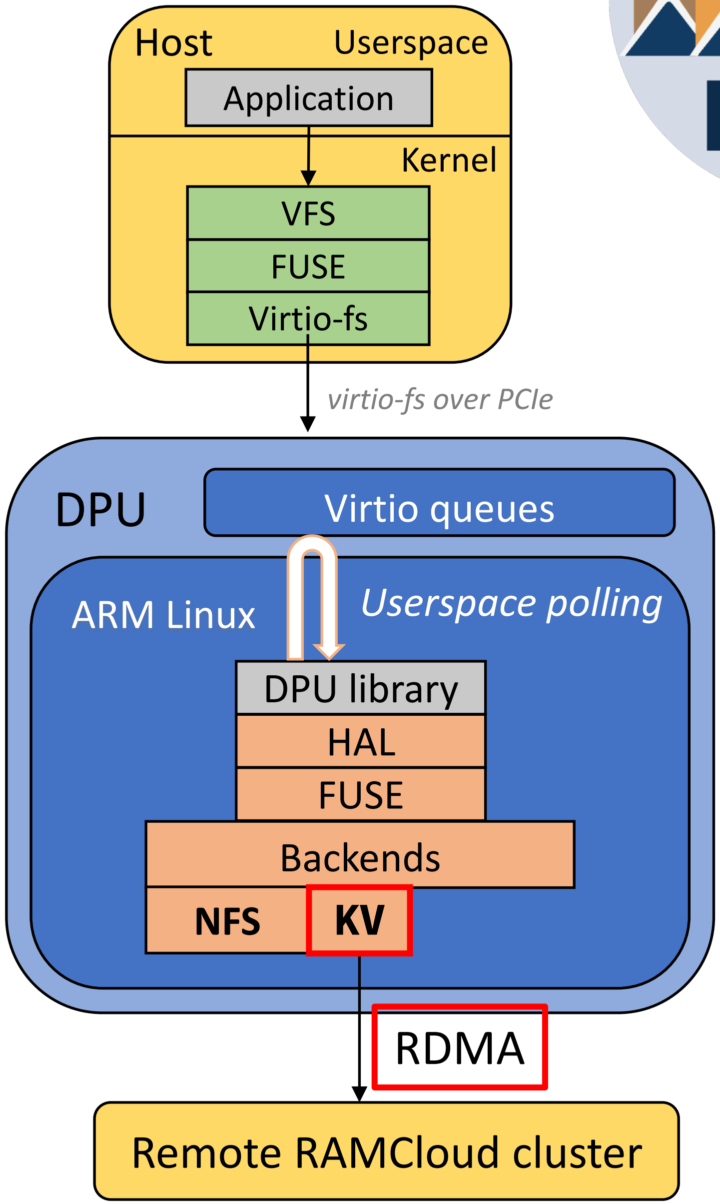
The Case for RAMClouds:
Scalable High-Performance Storage Entirely in DRAM

John Ousterhout, Parag Agrawal, David Erickson, Christos Kozyrakis, Jacob Leverich, David Mazières, Subhasish Mitra, Aravind Narayanan, Guru Parulkar, Mendel Rosenblum, Stephen M. Rumble, Eric Stratmann, and Ryan Stutsman

Department of Computer Science
Stanford University

Flat hierarchy

Optimized for 4k I/O and low latency



The **DPFS** framework: **D**PU-**P**owered **F**ile **S**ystems

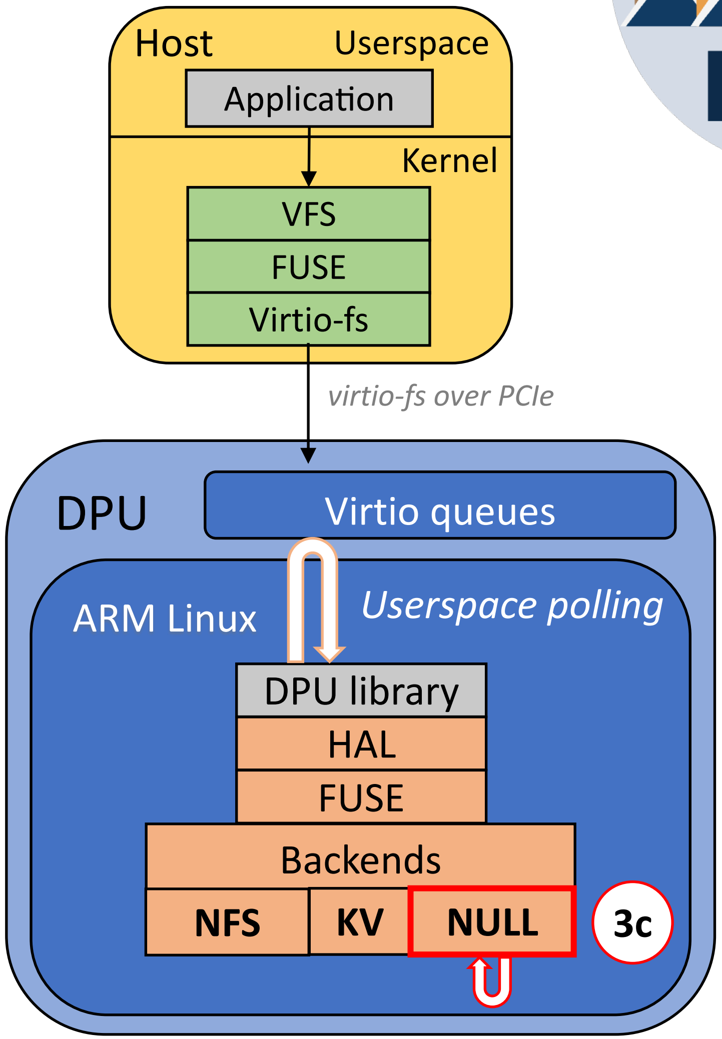
Architecture:

1. Hardware Abstraction Layer
2. FUSE API

3c **Several backends: NFS, KV, NULL**

Evaluates raw DPU performance:
latency and throughput

BlueField 2 vs BlueField 3 (soon)



Instantly returns any operation

Experimental evaluation

- Q1: Baseline performance when using a DPU (NULL)
- Q2: Throughput of DPFS-NFS (compared to Host NFS)
- Q3: Latency improvements with specialization (DPFS-NFS & -KV)
- Q4: Host CPU overhead analysis

Experimental setup

Host setup:

- 2x Intel Xeon E5-2630 v3, 2.2GHz, 8 cores/socket
- 128GiB DDR4 1600
- Clean Ubuntu 22.04 (Linux 6.2) and fio 3.28
- NFS with optimized settings per Google Cloud (does more caching than DPFS)

DPU:

- Nvidia BlueField-2
- 8x A72 ARM cores (running Ubuntu 20.04 Linux)
- 16GB single-channel DDR4
- 100Gb/s ConnectX-6 network interface
- Exposes a single virtio-fs device to a single bare metal host

Q1: Baseline DPFS performance (NULL)

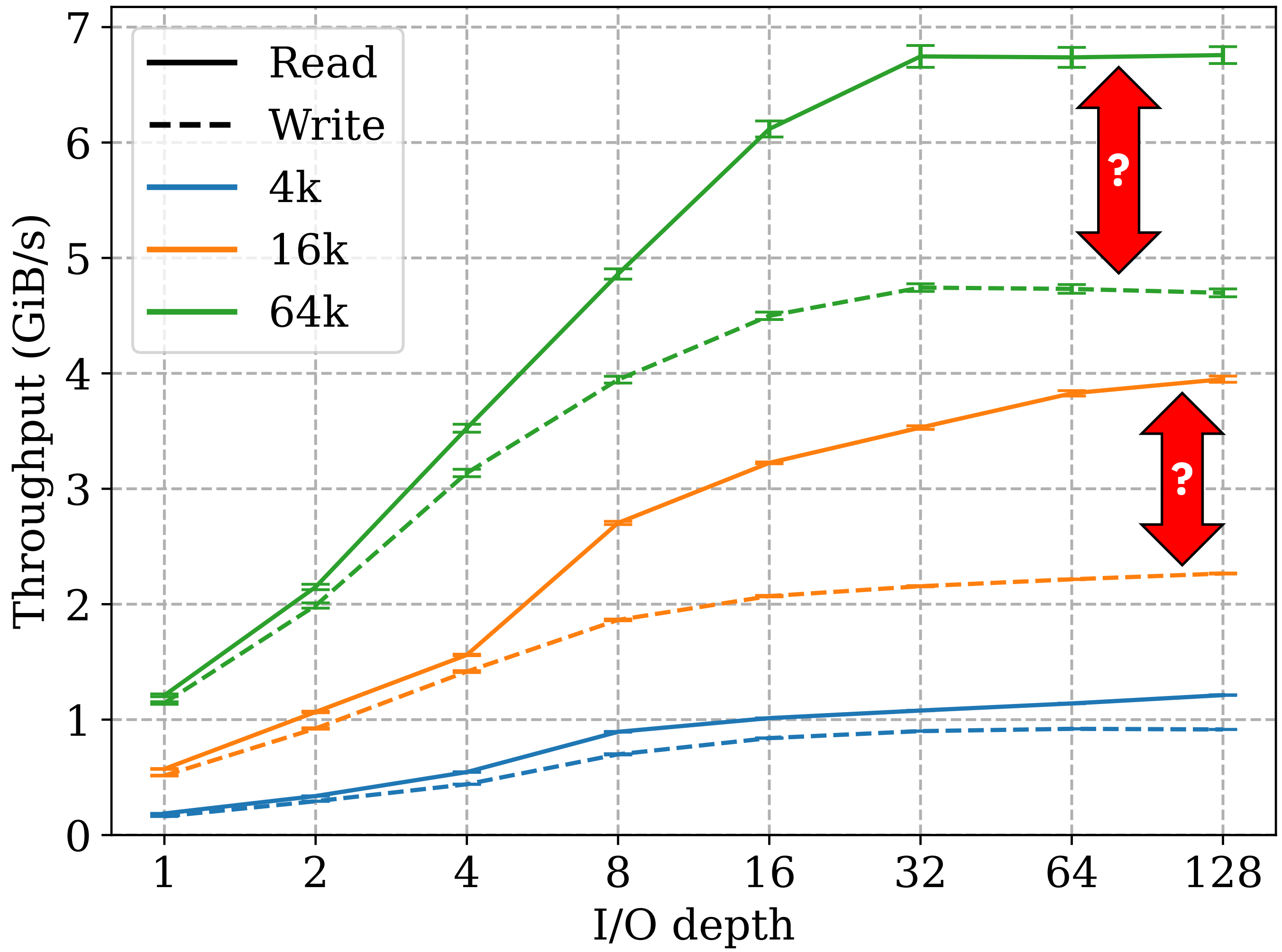
DPU setup:

- 1024 queue depth on the DPU
- Single core

Max TP = 7GB/s read and 5GB/s write
Large block sizes preferred

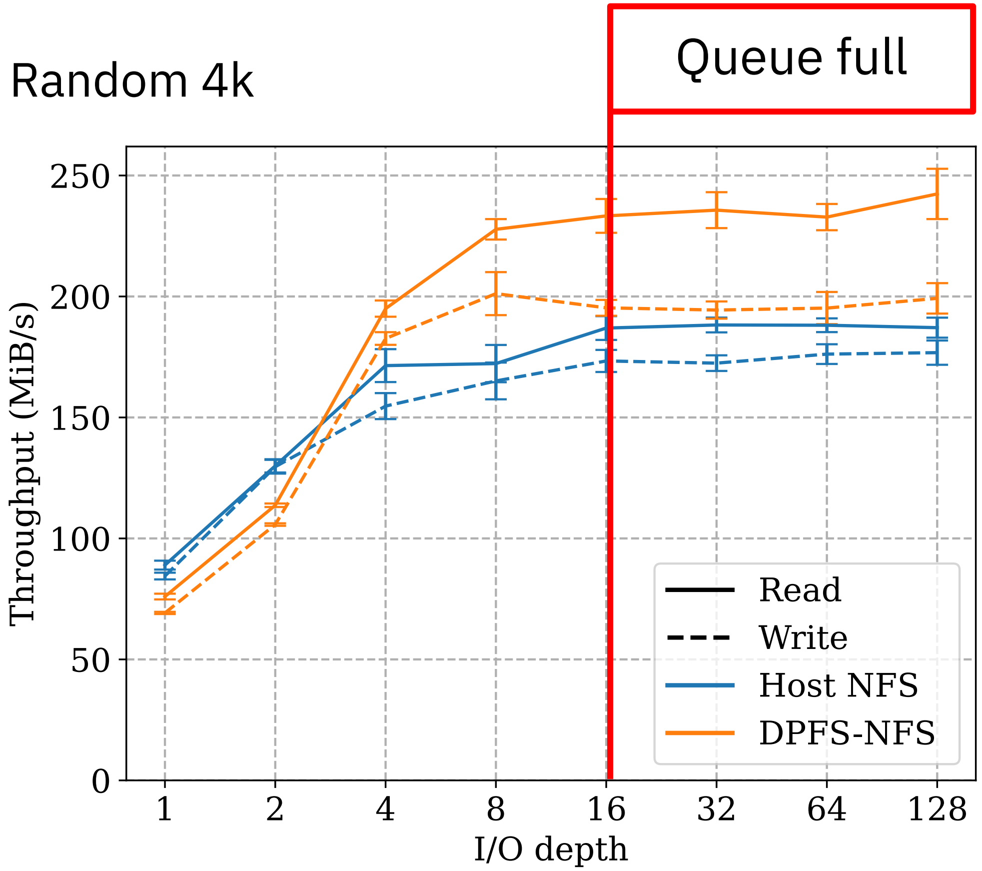
Read latency = 38.6μs
Write latency = 43.3μs
~40μs

Slow Arm A72 core fully saturated

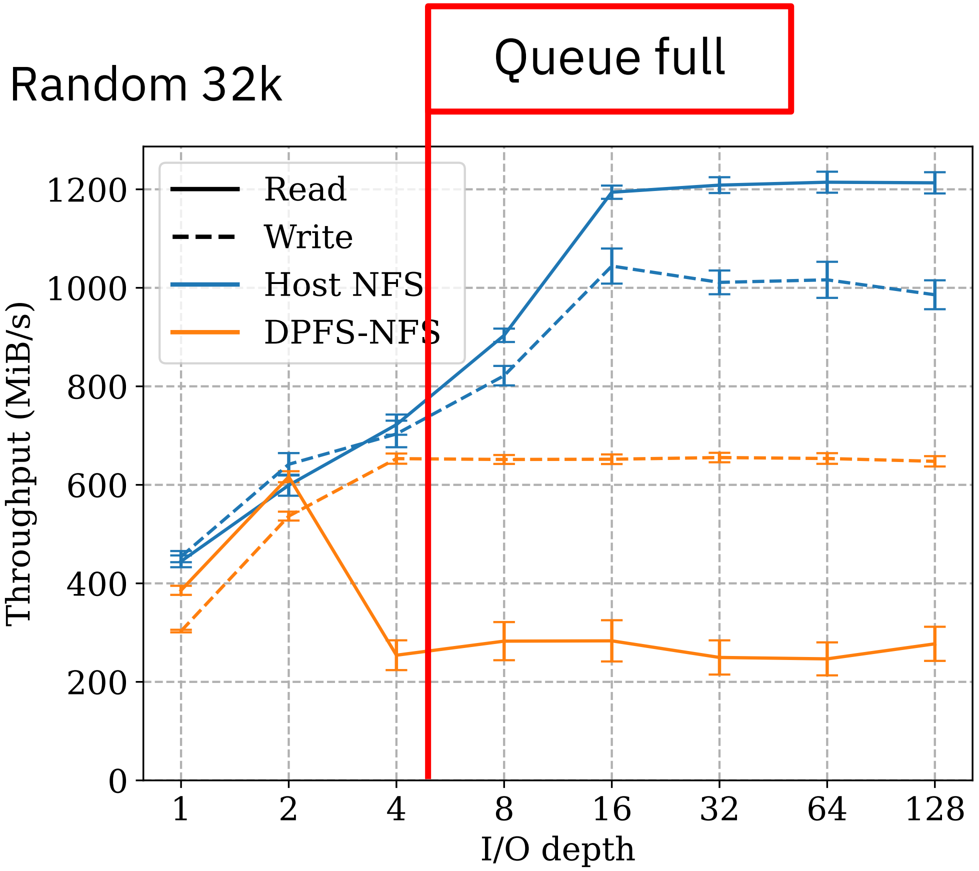


Q2: DPFS-NFS evaluation

Random 4k



Random 32k

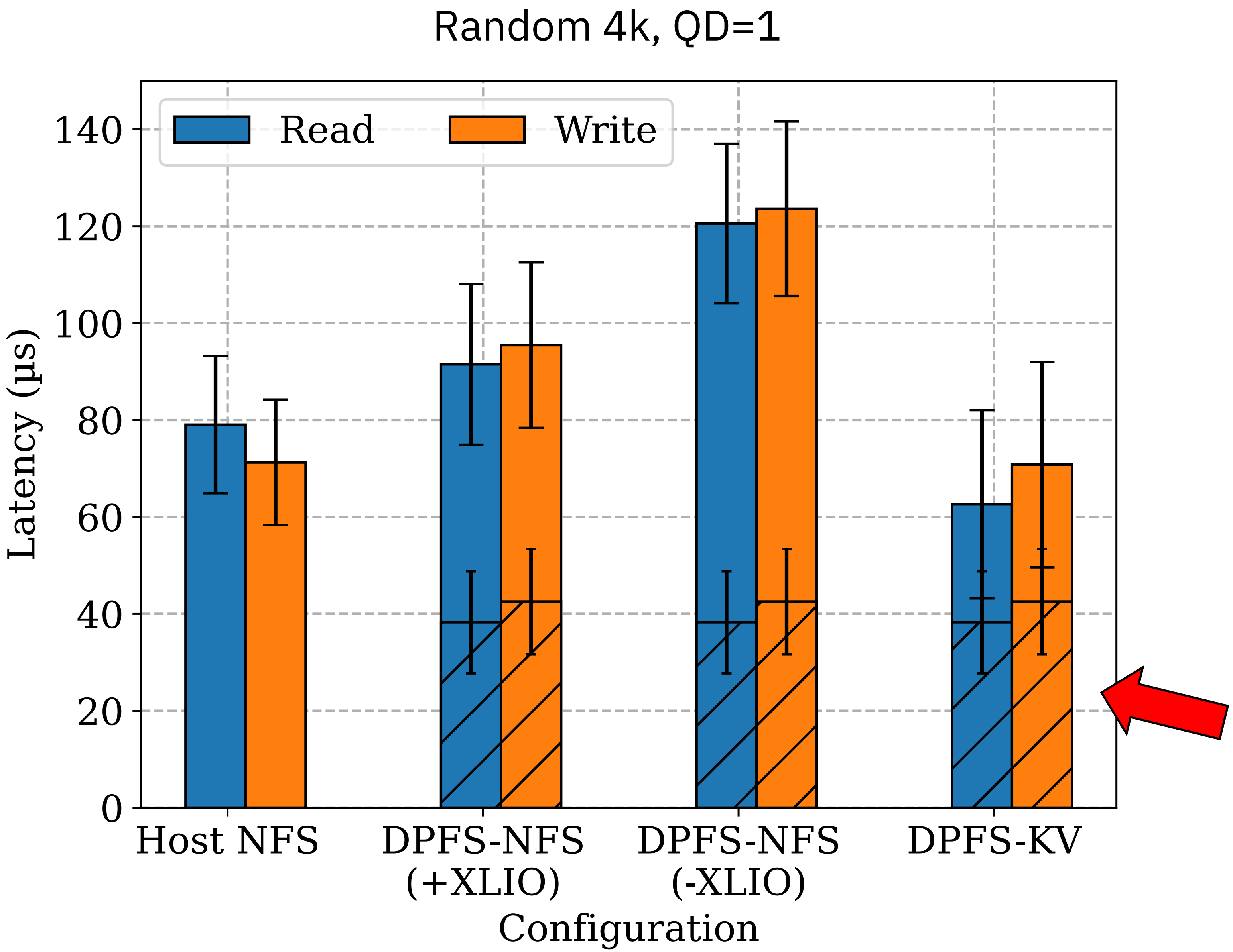


Bottleneck = TCP NFS I/O

Bottleneck = Limited queue depth (XLIO)

XLIO Read path *bad* with large BS & QD>=4

Q3: Latency evaluation



Hardware specialization is key (e.g. TCP offloading or RDMA)

Baseline DPFS-NULL latency

Q4: Evaluation CPU savings

Hypothesis:

Virtio-fs much lighter than NFS, so we expect big CPU savings.
(13k LoC vs 181k LoC)

Test setup:

- System-wide (kernel only) performance counters to account for RX path
- Take a 300s baseline, then perform a 300s stress test. Subtract the baseline from the stress test to only leave the instructions used for I/O.
- 4 KiB 50/50 read/write workload

	NFS	DPFS-NFS	+/-
Instructions/op	88,453	32,907	-62.80%
IPC	0.57	0.94	+64.21%
Branch miss rate	2.02	1.06	-47.42%
L1 dCache miss rate	8.82	3.82	-56.65%
dTLB miss rate	0.14	0.15	+7.14%
Savings in CPU cycles/op		**4.4×**	

Conclusions

- DPFS: a DPU-Powered File System Virtualization framework
  - Designed to meet the cloud FS needs of efficiency, management & security
  - 4.4x host cycle savings and similar performance to NFS
  - Multiple backends: NFS, NULL and KV

More info about the project at:

github.com/IBM/DPFS

Future work for DPFS

- Performance optimizations
  - *io_uring* file system backend for DPFS (DPU-local mirror)
  - Thread pooling in DPFS
  - Multi-queue support in virtio-fs and DPFS
  - Transition to faster DPUs (i.e., Nvidia BlueField-3)
- Multi-tenancy performance evaluation
- New RPC-based Virtio-fs backend
  - Split metadata and data paths, cut network hops and memory copies for data path

Thank you!

Info and contact about the project:

github.com/IBM/DPFS

Paper accepted at SYSTOR 2023!
(available 1st week of June)