@Large Research
Massivizing Computer Systems

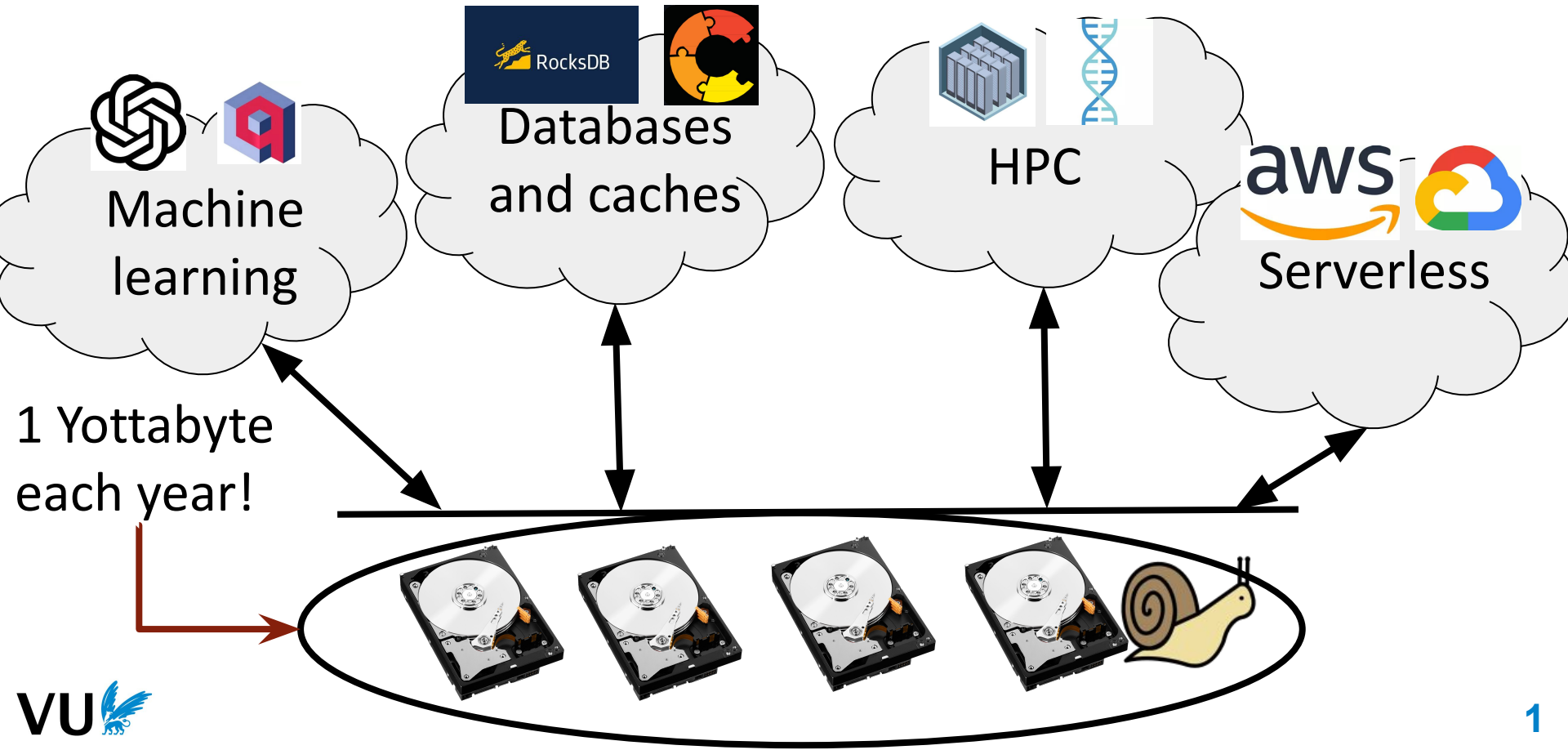
VU VRIJE UNIVERSITEIT AMSTERDAM

Does Linux Provide Performance Isolation for NVMe SSDs? Configuring cgroups for I/O Control in the NVMe Era

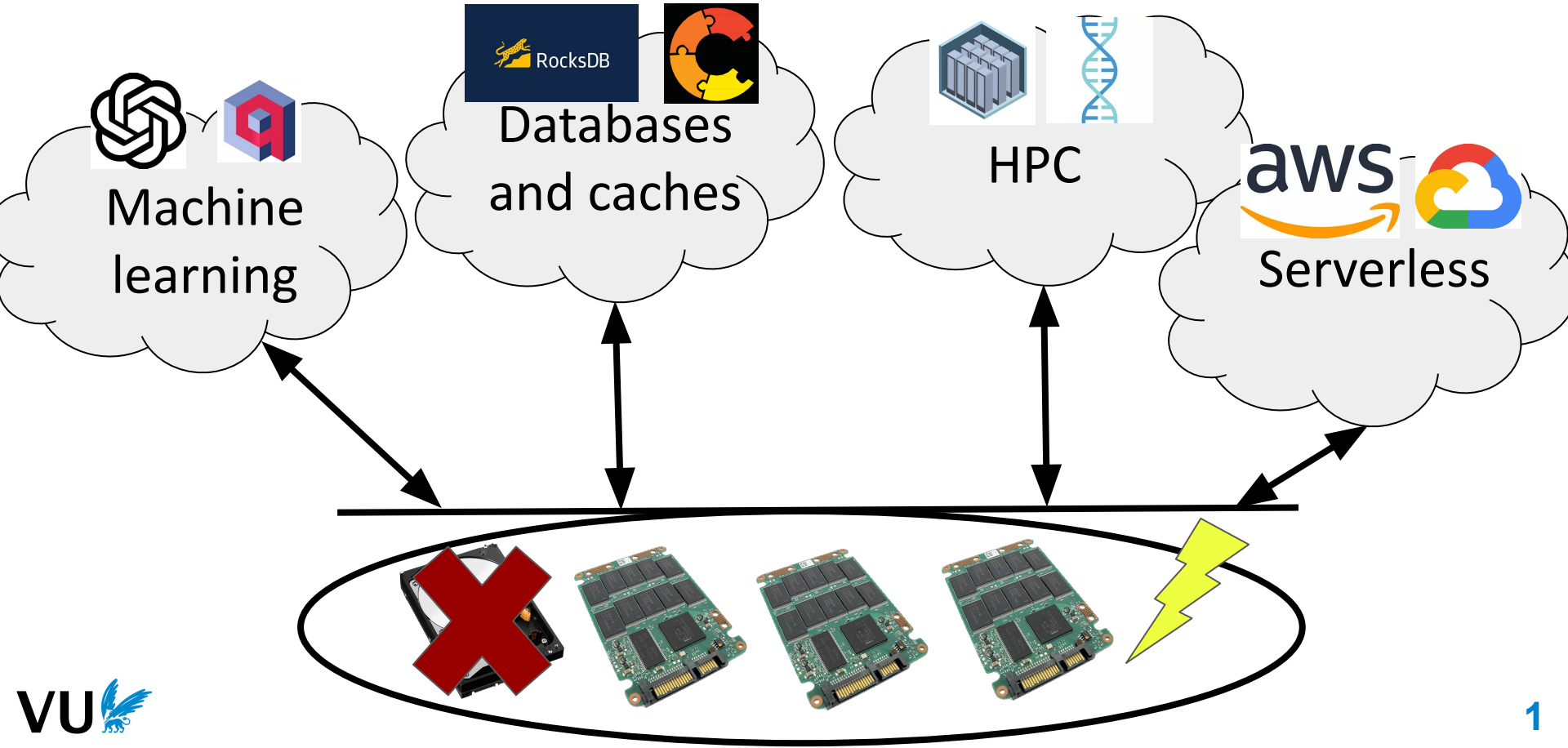
Krijn Doekemeijer, Zebin Ren, Tiziano De Matteis, Balakrishnan Chandrasekaran, and Animesh Trivedi

https://krien.github.io/

The amount of data is ever-increasing



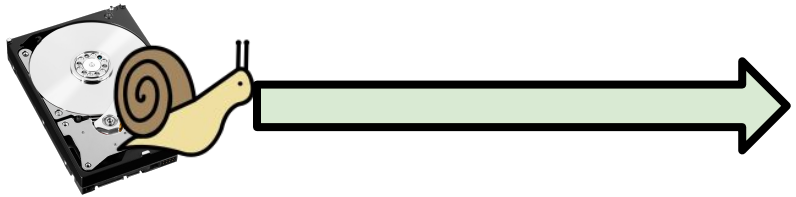
Data centers use **NVMe SSDs** for fast storage



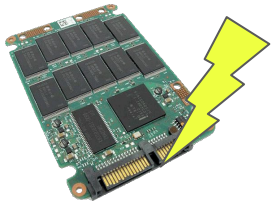
How fast is NVMe again?

Key point: NVMe SSDs obsolete the decades-long rule that storage is slow.

HDD



NVMe SDD

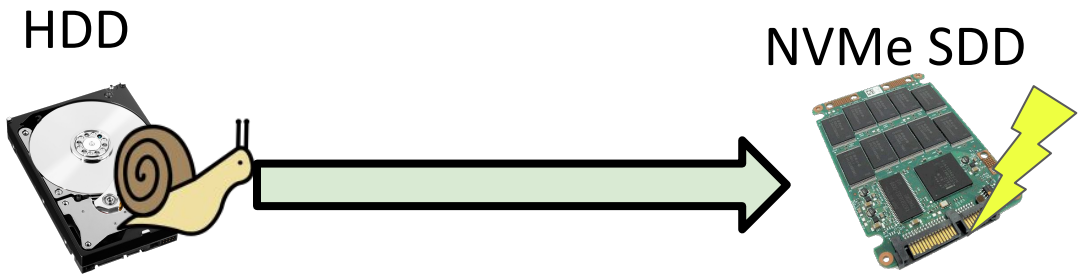


Latency: ms
Bandwidth: KiB/s

Latency: us
Bandwidth: GiB/s

How fast is NVMe again?

Key point: NVMe SSDs obsolete the decades-long rule that storage is slow.



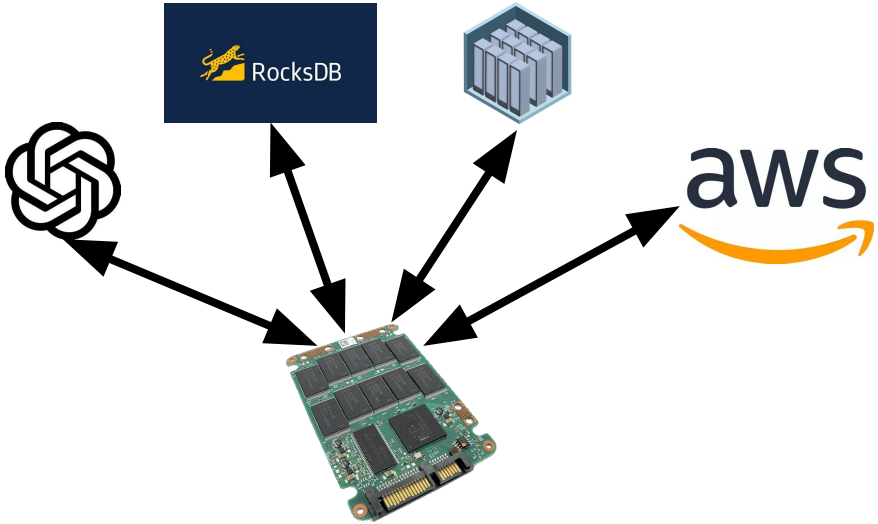
Latency: ms
Bandwidth: KiB/s

Latency: us
Bandwidth: GiB/s

Problem: OS software is designed for slow storage and generally has **high overhead** as a result.

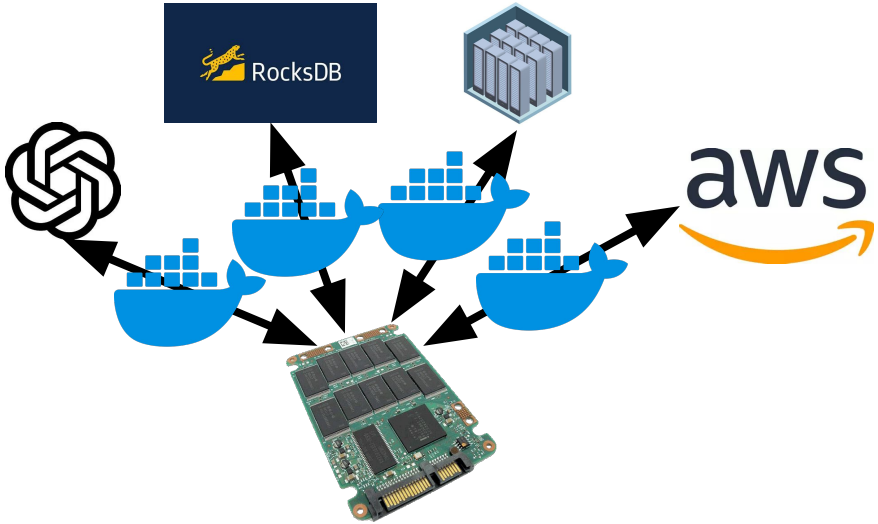
Data centers **share** NVMe storage resources

Key point: NVMe SSDs are shared by many workloads concurrently.



Data centers **share** NVMe storage resources

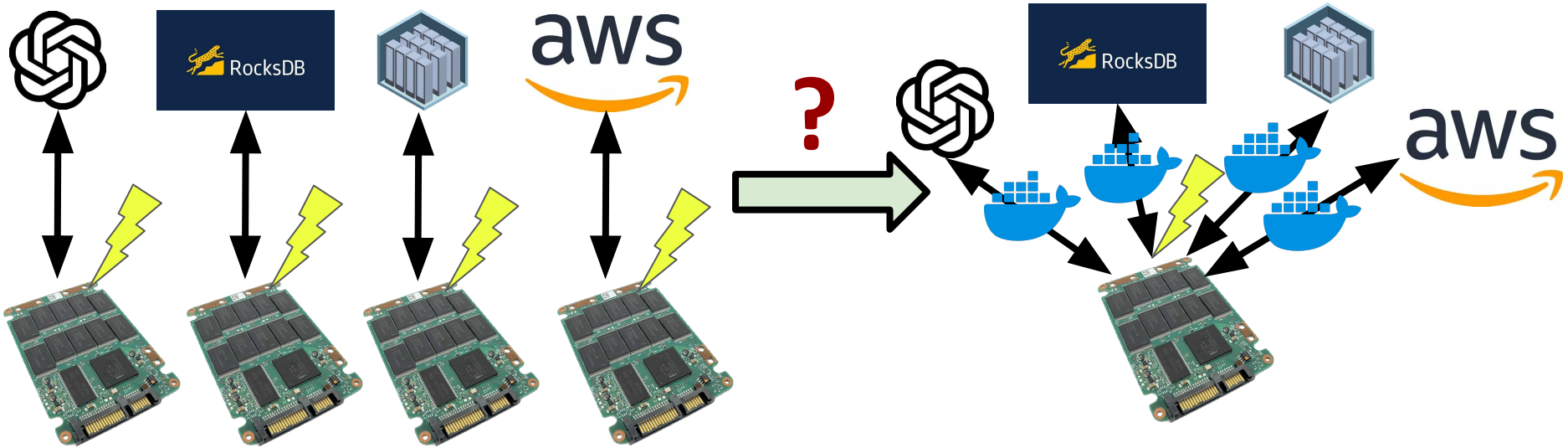
Key point: NVMe SSDs are shared by many workloads concurrently.



Problem: There is a need for **performance isolation** between workloads.

Key problem addressed in this work

Can Linux enable **low overhead performance isolation** for NVMe SSDs out of the box?



Open problems and contributions

1. What is storage performance isolation?
2. How to measure performance isolation?
3. What level of performance isolation is Linux capable of?

Open problems and contributions

1. What is the definition of storage performance isolation?
   - ➢ A unified definition of storage performance isolation (survey)
2. How to measure performance isolation?
3. What level of performance isolation is Linux capable of?

Open problems and contributions

1. What is the definition of storage performance isolation?
   - ➢ A unified definition of storage performance isolation (survey)
2. How to measure performance isolation?
   - ➢ isol-bench, a performance isolation benchmark suite
3. What level of performance isolation is Linux capable of?

Open problems and contributions

1. What is the definition of storage performance isolation?
   - ➢ A unified definition of storage performance isolation (survey)
2. How to measure performance isolation?
   - ➢ isol-bench, a performance isolation benchmark suite
3. What level of low overhead performance isolation is Linux capable of?
   - ➢ Evaluation and configuration exploration of state-of-the-practice cgroups

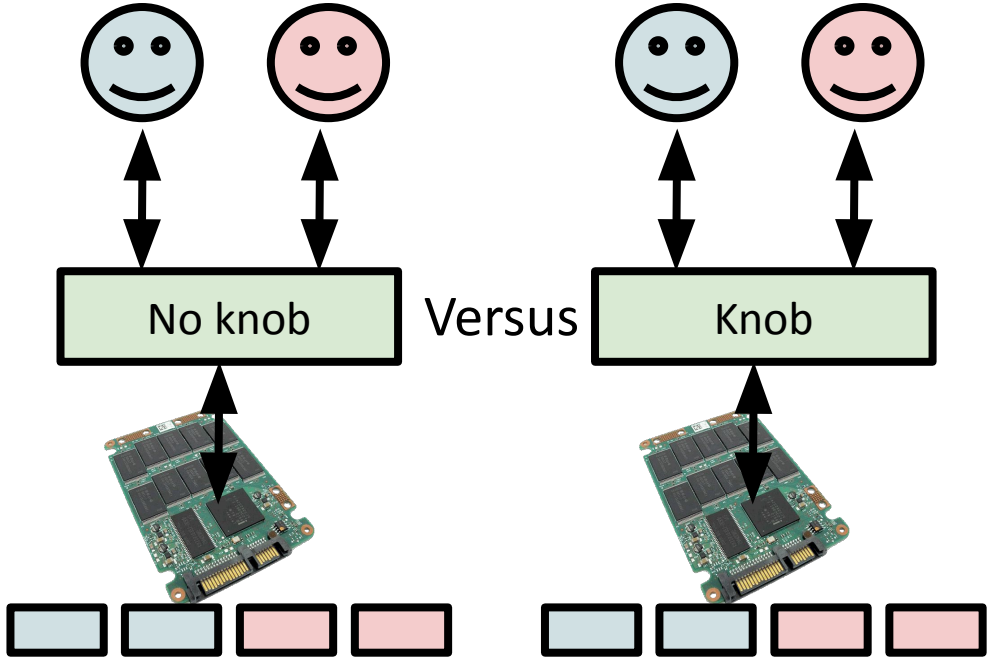
RQ1: What is storage performance isolation?

Using a survey of academic literature, we find a unified definition of isolation:

RQ1: What is storage performance isolation?

Using a survey of academic literature, we find a unified definition of isolation:

We explain isolation using the following illustration, where two tenants share an SSD's 4 bandwidth shares

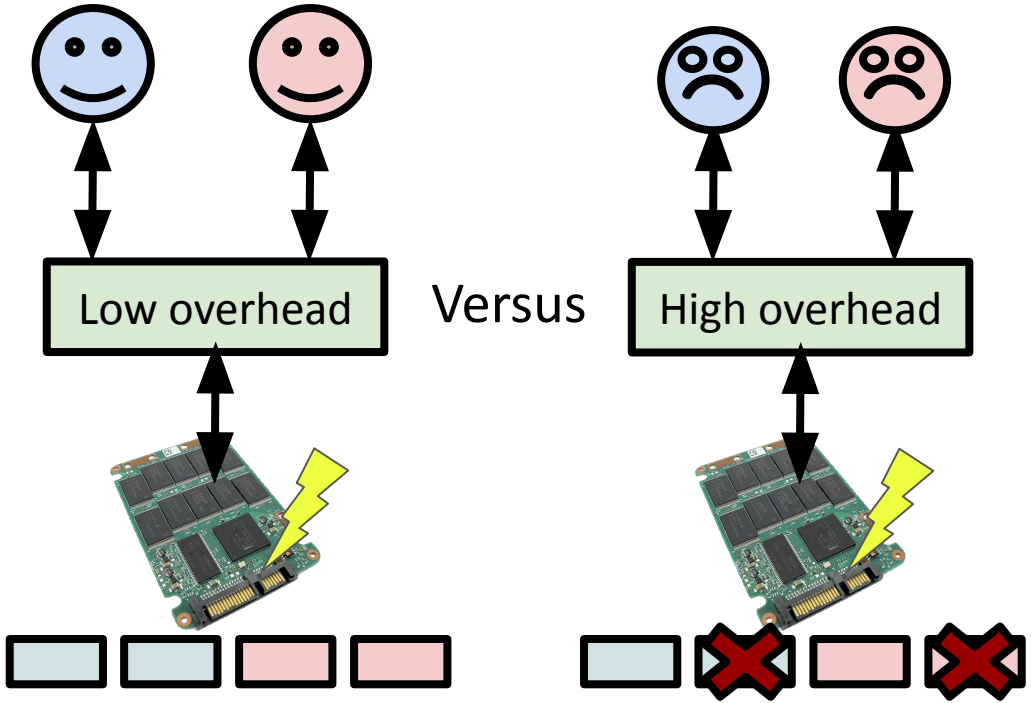


RQ1: What is storage performance isolation?

Using a survey of academic literature, we find a unified definition of isolation:

1. **Low overhead**

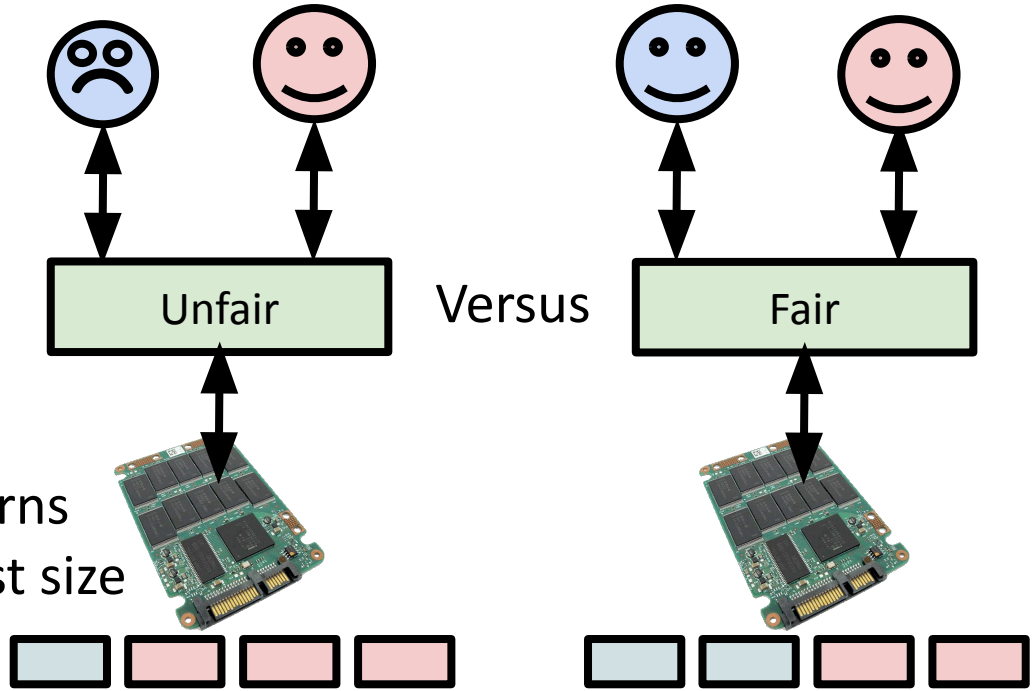
   ➢ latency, scalability, CPU



RQ1: What is storage performance isolation?

Using a survey of academic literature, we find a unified definition of isolation:

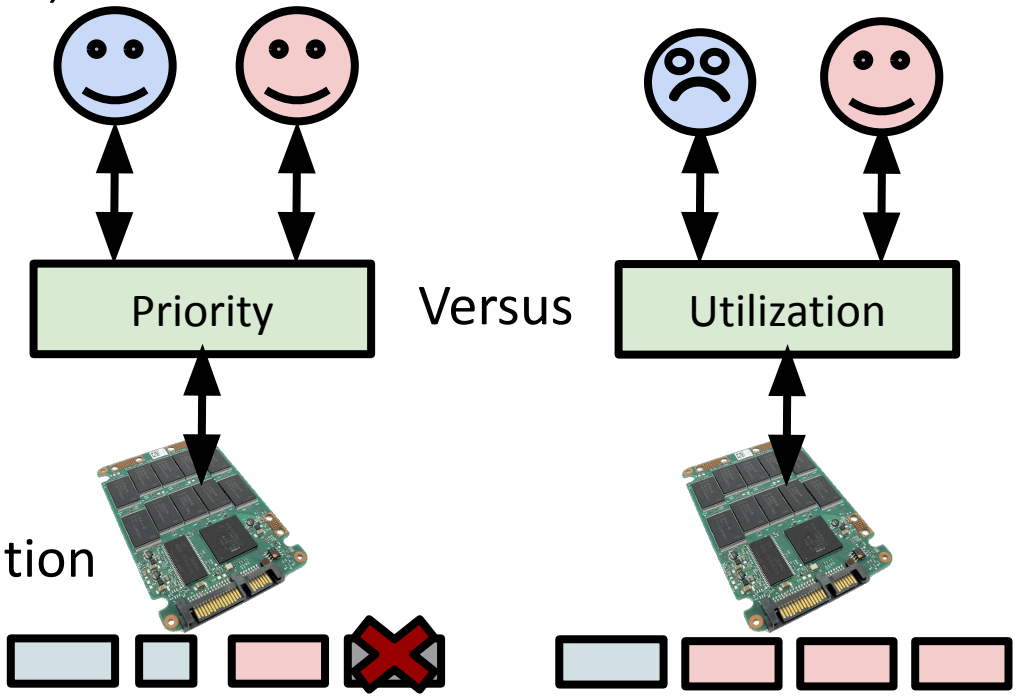
1. Low overhead
2. **Proportional fairness**
   - ➢ Equal bandwidth
   - ➢ Important if access patterns are different, e.g., request size
   - ➢ Jain's fairness index



RQ1: What is storage performance isolation?

Using a survey of academic literature, we find a unified definition of isolation:

1. Low overhead
2. Proportional fairness
3. **Priority utilization trade-offs**
   - ➢ Priority tenants
   - ➢ Administrator wants utilization
   - ➢ How to trade-off priority against utilization?



RQ1: What is storage performance isolation?

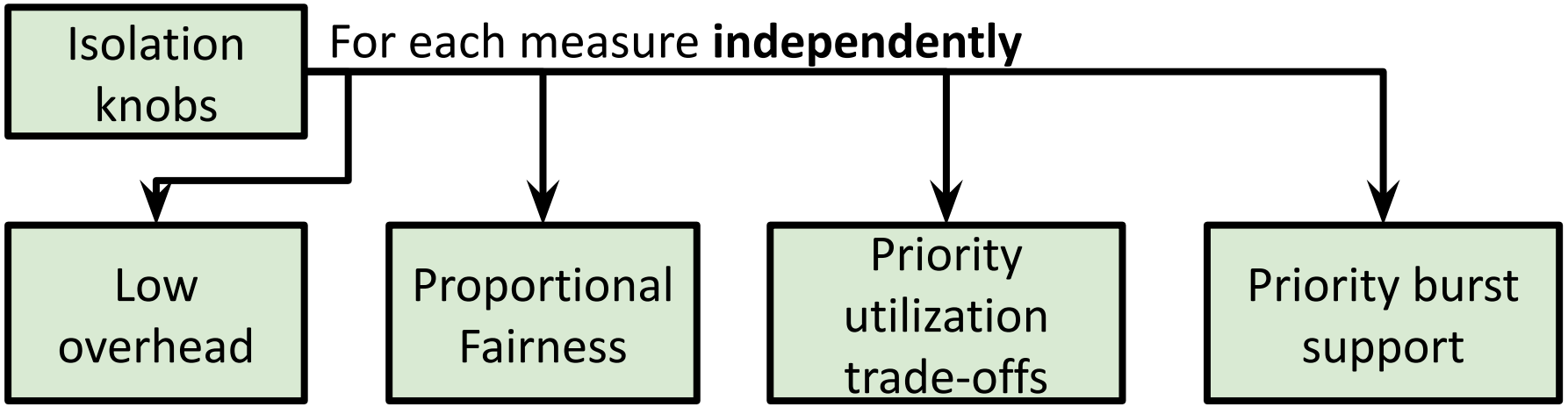
Using a survey of academic literature, we find a unified definition of isolation:

1. Low overhead
2. Proportional fairness
3. Priority utilization trade-offs
4. **Burst support**

➢ Many workloads, e.g., serverless, are short running (ms)

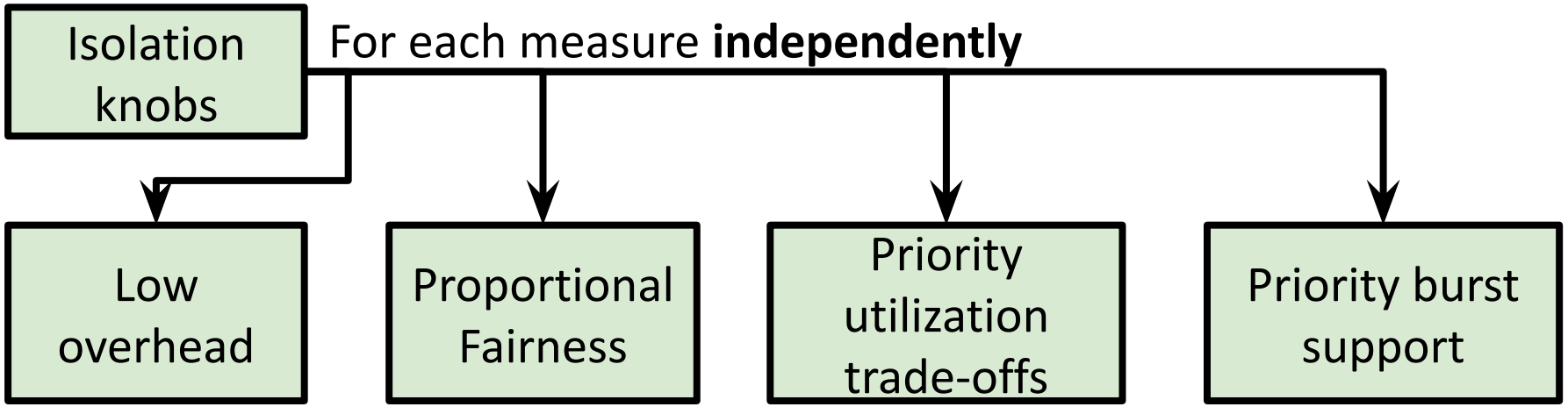
RQ2: How to measure performance isolation?

We introduce **isol-bench**, a benchmark suite



RQ2: How to measure performance isolation?

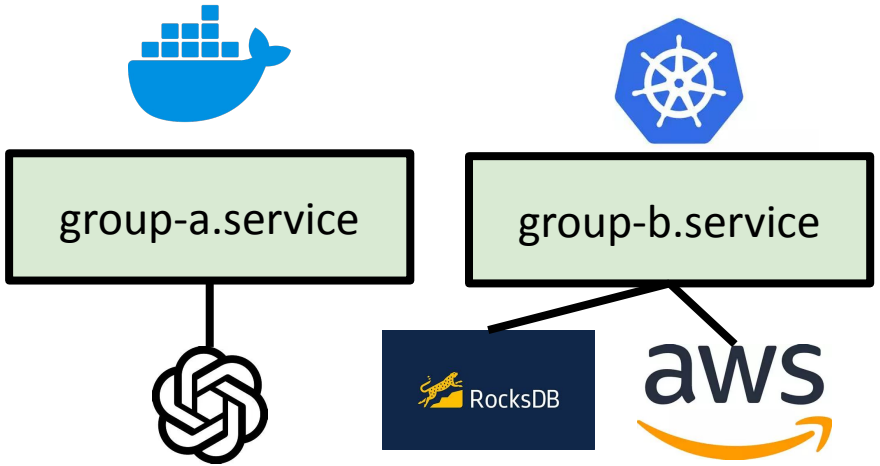
We introduce **isol-bench**, a benchmark suite



Important: not a single metric is returned as **operator needs** differ!

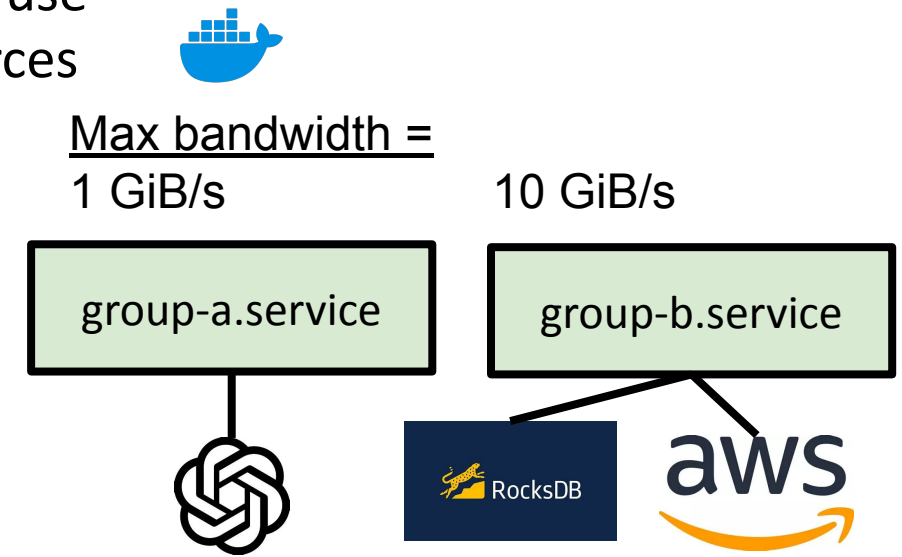
Background: cgroups - Linux I/O control

- In Linux all processes are part of a cgroup
- A cgroup, or control group, controls the resources its collective processes can use



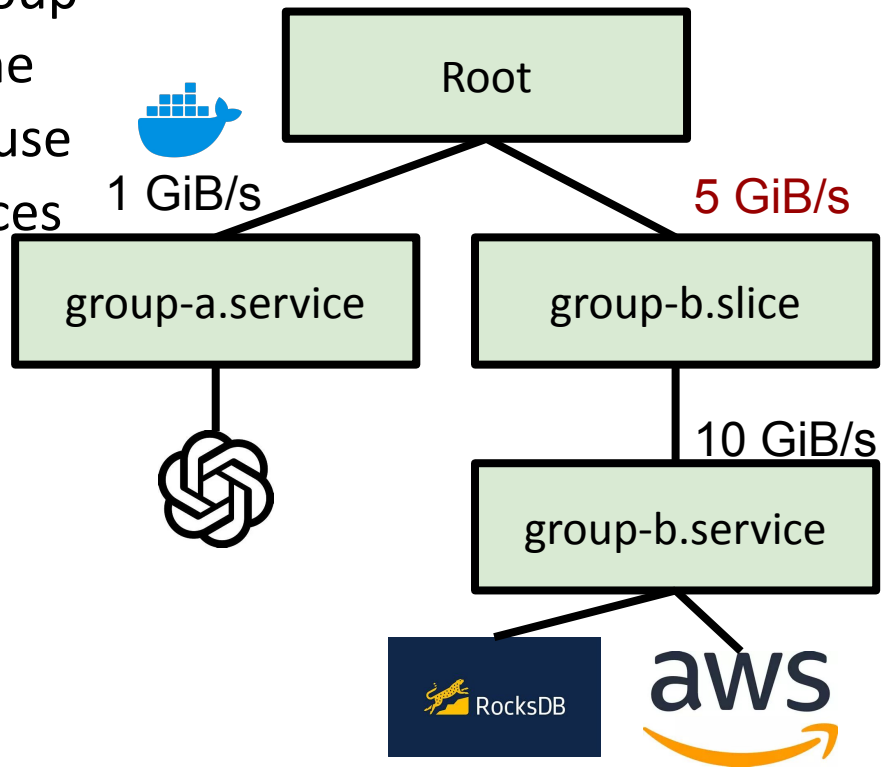
Background: cgroups - Linux I/O control

- In Linux all processes are part of a cgroup
- A cgroup, or control group, controls the resources its collective processes can use
- I/O control knobs limit storage resources
  - e.g. io.max



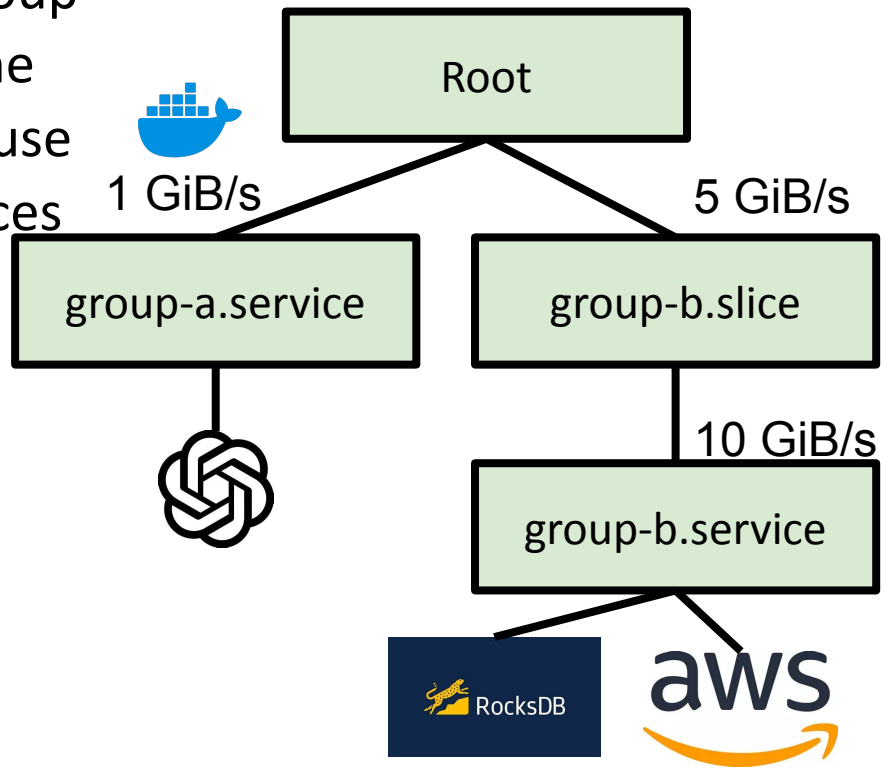
Background: cgroups - Linux I/O control

- In Linux all processes are part of a cgroup
- A cgroup, or control group, controls the resources its collective processes can use
- I/O control knobs limit storage resources
  - e.g. io.max
- cgroups is hierarchical



Background: cgroups - Linux I/O control

- In Linux all processes are part of a cgroup
- A cgroup, or control group, controls the resources its collective processes can use
- I/O control knobs limit storage resources
- cgroups is hierarchical
- In total 5 knobs:
  1. io.max
  2. io.latency
  3. io.cost
  4. MQ-DL
  5. BFQ



RQ3: What performance isolation is Linux capable of?

Using isol-bench we evaluate all **four** isolation desiderata for **cgroups**

RQ3: What performance isolation is Linux capable of?

Using isol-bench we evaluate all **four** isolation desiderata for **cgroups,** i.e., fill in ?

cgroup knob	Low overhead	Proportional Fairness	Priority utilization trade-offs	Priority bursts
MQ-DL	?	?	?	?
BFQ	?	?	?	?
io.max	?	?	?	?
io.latency	?	?	?	?
io.cost	?	?	?	?

VU

RQ3: What performance isolation is Linux capable of?

Using isol-bench we evaluate all **four** isolation desiderata for **cgroups,** i.e., fill in ?

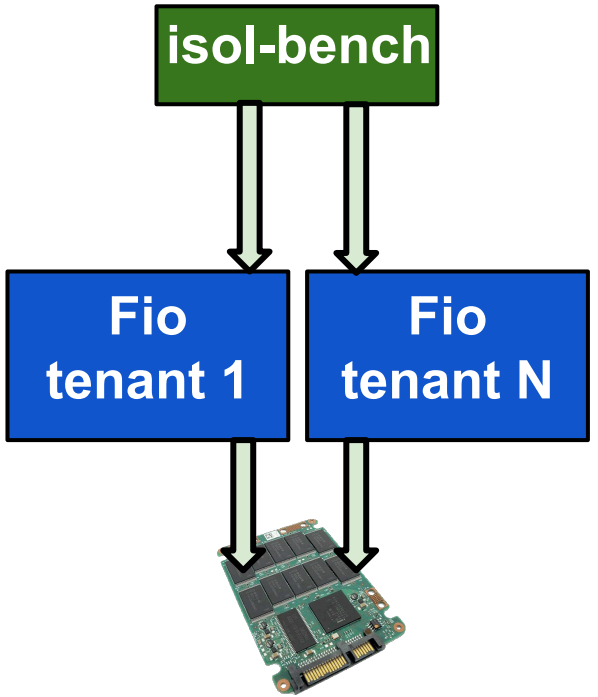
cgroup knob	**Low overhead**	**Proportional Fairness**	Priority utilization trade-offs	Priority bursts
MQ-DL	?	?	?	?
BFQ	?	?		?
io.max	?	?		?
io.latency	?	?	?	?
io.cost	?	?	?	?

We will discuss these two as examples

Question: Which knob provides most isolation?

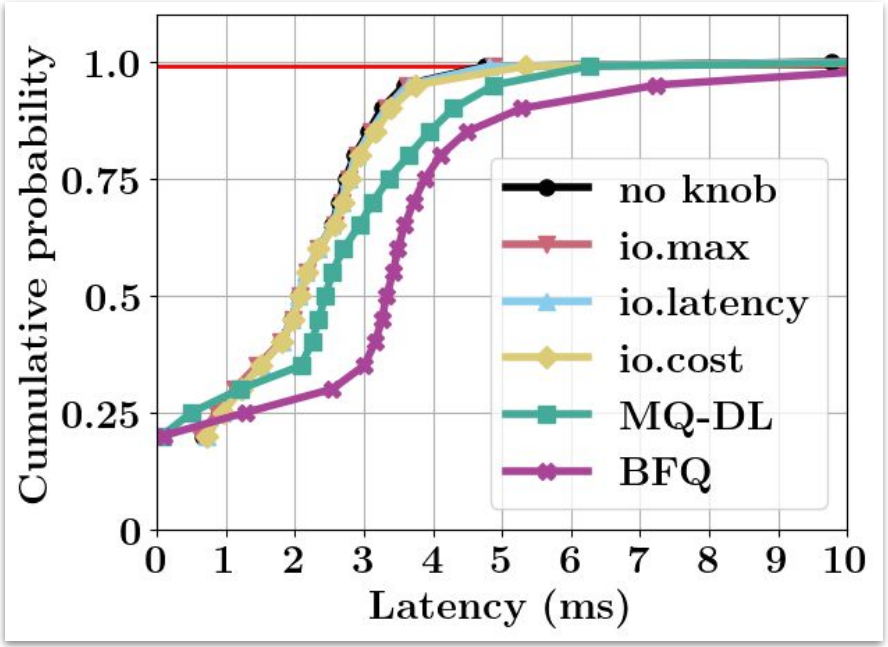
isol-bench setup for cgroups

- Workload generator: fio
  - Use one fio instance per client
- OS config: Linux 6.9, **DIRECT_IO, no file system**
- Run on 2 NVMe SSD models
  - 1–7 Flash medium: Samsung 980 pro
  - 1–7 Other medium: Intel Optane



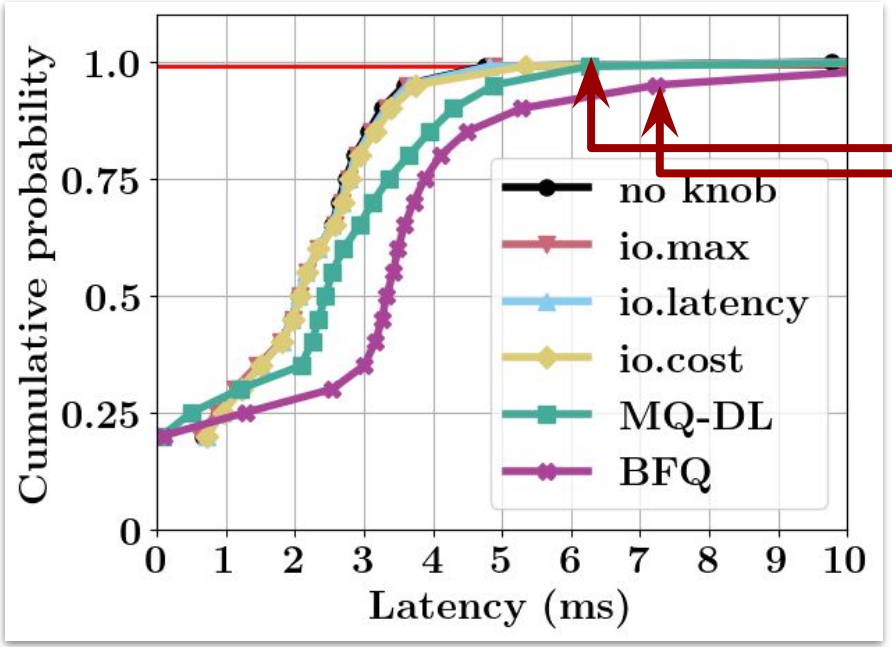
Desiderata 1: Low performance overhead

Latency CDF
(more to left is better)



Desiderata 1: Low performance overhead

Latency CDF
(more to left is better)



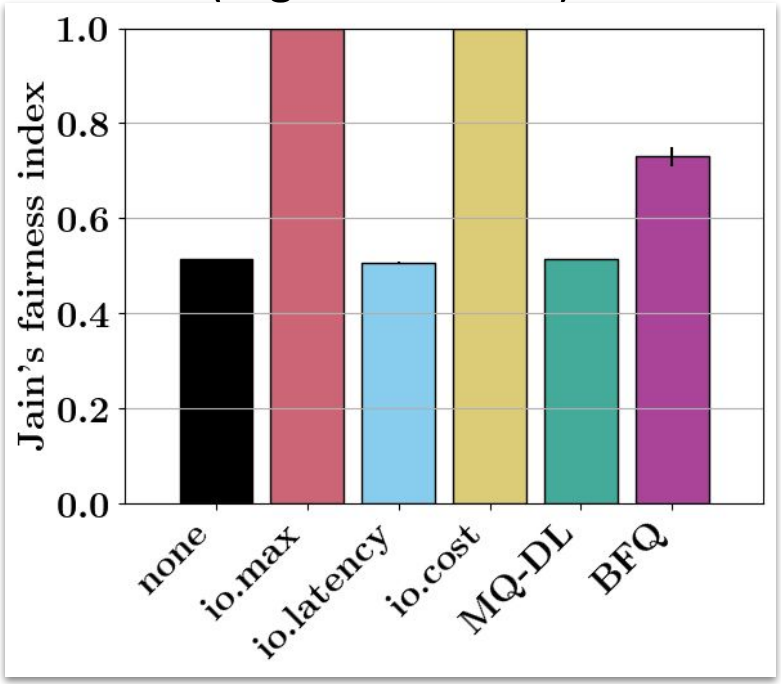
Schedulers have high overhead!

Desiderata 2: Fairness

- Workload characteristics affect fairness!
  - Request size
  - Access patterns
  - Mixed reads and writes
- We give an example of request size (4 KiB and 256 KiB mixed).

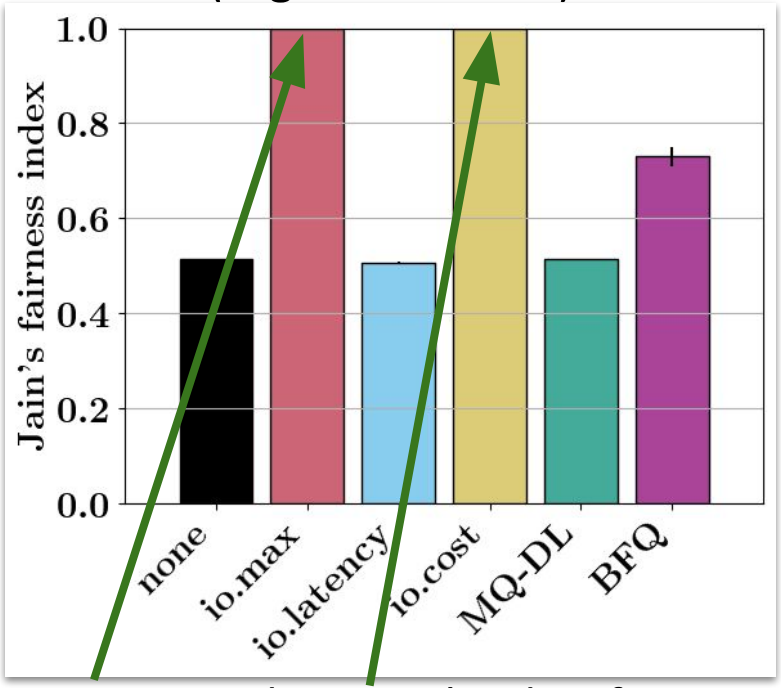
Desiderata 2: Fairness

Jain's fairness index for a mixed workload of 4 and 256 KiB
(Higher is better)



Desiderata 2: Fairness

Jain's fairness index for a mixed workload of 4 and 256 KiB
(Higher is better)



io.max and io.cost lead to fairness
(for reference, a fairness of 0.5 can be a 100+MiBs difference)

Overall results

Desiderata achieved per knob:

cgroup knob	Low overhead	Proportional Fairness	Priority utilization trade-offs	Priority bursts
MQ-DL	✗	✗	✗	✗
BFQ	✗	✗	✗	✗
io.max	✓	—	—	—
io.latency	✓	✗	—	✗
io.cost	—	✓	✓	✓

Overall results

Desiderata achieved per knob:

cgroup knob	Low overhead	Proportional Fairness	Priority utilization trade-offs	Priority bursts
MQ-DL	✗	✗	✗	✗
BFQ	✗	✗	✗	✗
io.max	✓	—	—	—
io.latency	✓	✗	—	✗
io.cost	—	✓	✓	✓

I/O schedulers were already confirmed to have high overhead. Here we confirm that they do not enable isolation either.

Overall results

Desiderata achieved per knob:

cgroup knob	Low overhead	Proportional Fairness	Priority utilization trade-offs	Priority bursts
MQ-DL	✗	✗	✗	✗
BFQ	✗	✗	✗	✗
io.max	✓	–	–	–
io.latency	✓	✗	–	✗
io.cost	–	✓	✓	✓

io.cost provides highest isolation, but occurs an overhead past CPU saturation.

What to do with our results

Our results show that practitioners:

1. I/O schedulers should be avoided for NVMe, even for isolation.
2. io.cost can be used for isolation instead.
    a. Meta already uses io.cost in production.

More details/results in the paper…

Does Linux Provide Performance Isolation for NVMe SSDs? Configuring cgroups for I/O Control in the NVMe Era

Krijn Doekemeijer
Department of Computer Science
Vrije Universiteit Amsterdam
Amsterdam, the Netherlands

Zebin Ren
Department of Computer Science
Vrije Universiteit Amsterdam
Amsterdam, the Netherlands

Tiziano De Matteis
Department of Computer Science
Vrije Universiteit Amsterdam
Amsterdam, the Netherlands

Balakrishnan Chandrasekaran
Department of Computer Science
Vrije Universiteit Amsterdam
Amsterdam, the Netherlands

Animesh Trivedi
IBM Research Europe
Zurich, Switzerland

***Abstract*—Modern storage workloads commonly run in containers within data centers, such as machine learning, databases, caches, HPC, and serverless workloads. To facilitate the storage performance requirements (e.g., bandwidth, latency) of these workloads, data centers have adopted fast NVMe SSDs as a storage medium. At the same time, data centers virtualize and share these storage resources with multiple tenants to improve resource utilization and reduce costs. Such sharing leads to an inherent trade-off between tenant performance isolation and SSD utilization. Although various research studies demonstrate how to achieve various performance isolation properties, such as fairness, there is neither a unified definition for performance isolation nor a benchmark. Furthermore, the isolation capabilities of state-of-the-practice I/O control mechanisms in the Linux kernel are not well understood. In this paper, we address these three challenges. First, we survey the definition of performance isolation and uncover four common performance isolation desiderata. Second, we introduce `isol-bench`, a benchmark for evaluating these desiderata for I/O control mechanisms. Third, we use `isol-bench` to evaluate I/O isolation for Linux's state-of-the-practice I/O control mechanism, cgroups. From our evaluation, we are able to conclude that out of cgroups's knobs `io.cost` achieves the most isolation desiderata, but has a latency overhead past CPU saturation. We open-source the source code of isol-bench at https://github.com/atlarge-research/isol-bench.**

***Index Terms*—cgroups, Measurements, NVMe, performance isolation**

I. Introduction

A large number of storage workloads run in containers within data centers, including machine learning [64], databases [94], caches [77], HPC [43], and serverless applications [50], [93]. To facilitate the performance requirements of such containerized workloads, data centers have adopted fast NVMe SSD storage with single-digit microsecond latencies, millions of IOPS of throughput, and GiB/s of bandwidth [38], [78]. However, since SSD resource utilization by workloads is typically low [71], [100], storage resources are virtualized and shared between containerized tenants to improve SSD utilization and prevent resource stranding [64], [71], [73], [84], [100]. Sharing leads to an implicit trade-off between SSD utilization and tenant performance isolation; hence, various works investigate how to co-locate tenants while isolating tenant performance [34], [53], [83]. Nevertheless, there is limited understanding of the structural definition of performance isolation for storage and a common benchmark to compare isolation capabilities. In this work, we address these shortcomings by surveying storage performance isolation to understand and summarize the community's understanding of isolation, and by benchmarking the isolation capabilities of Linux's cgroups I/O control knobs.

The first challenge we address is the lack of a unified definition and common benchmark for storage performance isolation, i.e., research studies use different properties and metrics to evaluate isolation. For example, some works consider isolation as minimizing tail latency for priority workloads [61], whereas others focus on fairness [82]. In short, there is no unified definition of storage performance isolation. This lack of definition limits the effectiveness of isolation efforts and leads to apples-to-pears comparisons when comparing solutions such as the various knobs exposed in cgroups. To address this challenge, we take a two-pronged approach. First, we define isolation by summarizing the state-of-the-practice isolation desiderata using a survey. From this survey, we derive four performance isolation desiderata, which we discuss in detail in §II. Second, to effectively compare isolation solutions, we propose `isol-bench`, a benchmark suite that evaluates all our isolation desiderata for a given system.

The next challenge we address is that the performance isolation capabilities of state-of-the-practice I/O control knobs on NVMe SSDs are unknown for Linux. Enabling isolation properties for NVMe SSDs on the host is challenging because (i) I/O workloads are highly dynamic [74]; (ii) NVMe SSDs have high performance requiring low CPU overhead for I/O control [91]; (iii) SSD performance models differ significantly [54]; and (iv) the SSD-internal flash medium has various performance idiosyncrasies [54] (e.g., asymmetric read and write performance, garbage collection). Various state-of-the-art solutions have been proposed to improve SSD isolation. Such works typically utilize hardware-specific isolation, requiring specialized SSDs such as open-channel SSDs [17], [72], [83], or employ software isolation, typically running in user-space or modified kernels [37]. However, access to specialized hardware is often infeasible due to costs or availability, and custom software solutions require domain expertise; hence, it is equally vital to understand what is already available in the Linux kernel by default. In this work, we use `isol-bench` to benchmark the isolation capabilities of the state-of-the-practice cgroups.

The state-of-the-practice for containers in Linux is to use platforms such as Docker, LXC/LXD, or Podman, where (virtual) SSD resources are managed with Linux' cgroups [19], [34]. cgroups provides various knobs for I/O control; for example, to limit bandwidth [5], to prioritize workloads, or to do weighted sharing [66], [84]. `isol-bench` evaluates all our survey-defined performance isolation desiderata for cgroups using the *fio* workload generator [14]. We evaluate all I/O control cgroups knobs, which are `io.prio.class`, `io.bfq.weight`, `io.max`, `io.latency`, `io.cost`, and `io.weight`. We present our key findings in Tab. I. Note that some knobs such as `io.prio.class`, `io.bfq.weight` or `io.weight` require I/O schedulers or other cgroups knobs to be active to have a performance effect; hence, we list these as combinations in the table. From our analysis, we report that `io.cost` achieves the highest level of performance isolation on NVMe albeit with a small latency overhead beyond CPU saturation.

We summarize our key contributions as follows:

- A survey on the definition of performance isolation for data center storage, where we summarize the definition to include performance overhead and scalability, proportional fairness, trade-offs between prioritization and utilization, and priority burst support.
- `isol-bench`, a benchmarking suite for storage performance isolation, which we use to evaluate the isolation capabilities of cgroups I/O control knobs.
- A first-of-its-kind study of performance isolation for cgroups on NVMe SSDs, where, through 10 observations, we determine that `io.cost` achieves most of the isolation desiderata, depending on its configuration.
- To facilitate reproduction, we open-source design and implementation of our code as FAIR data sets at https://github.com/atlarge-research/isol-bench.

TABLE I: *Performance isolation desiderata for cgroups; the "+" indicates that a knob had to be evaluated together with an I/O scheduler (i.e., `MQ-DL`, `BFQ`) or other knob.*

cgroups I/O control knob	*Low Overhead*	*Proportional Fairness*	*Priority Utilization Trade-offs*	*Priority Bursts*
`io.prio.class` + `MQ-DL`	✗	✗	✗	✗
`io.bfq.weight` + `BFQ`	✗	✗	✗	✗
`io.max`	✓	—	—	—
`io.latency`	✓	✗	—	✗
`io.cost` + `io.weight`	—	✓	✓	✓

This work is partially supported by Netherlands-funded projects NWO OffSense (OCENW.KLEIN.209), NWO MLS (OCENW.KLEIN.561), and GFP 6G FNS. The work is also supported by EU-funded projects MSCA CloudStars (g.a. 101086248) and Horizon Graph Massivizer (g.a. 101093202). Krijn Doekemeijer is funded by the VU PhD innovation program.

II. Survey On Storage Performance Isolation

In this section, we survey the definition of performance isolation in the context of (NVMe) SSDs. With this survey, we aim to address two key challenges: there is currently a lack of understanding of the broader definition of storage performance isolation and a lack of a methodology for evaluating such performance isolation. Our survey addresses these problems by distilling a usable isolation definition that can be benchmarked and quantified; specifically, we discuss tenants' performance requirements, and ubiquitous performance isolation desiderata and how to quantify them.

A. Tenant Performance Requirements

To evaluate performance isolation, we first need to understand tenant requirements in terms of performance. In the literature, we observe that such requirements are commonly defined in terms of *service-level objectives* (SLOs) [13], [20], [27], [30], [31], [35], [36], [40], [45], [46], [49], [55], [59], [65], [69], [93], [99], [101]. Common SLOs include throughput and bandwidth (average, minimum, maximum), latency/slowdown (average, P99 tail, minimum, maximum, CDF), and burstiness for throughput and bandwidth. Here, throughput refers to operations per second and bandwidth refers to I/O bytes per second read from or written to the SSD. Furthermore, in the cloud, many solutions allow selecting a performance profile [1]–[3], [6], [7], e.g., provisioned throughput/bandwidth and average/P99 latency. AWS EBS, for example, has "provisioned iops," which gives a non-guaranteed approximation of expected throughput [2]. Such profiles are generally tied to volume size and are neither application-defined nor guaranteed.

In this work, we do not use SLOs or profiles; instead, we use the metrics they represent. Practitioners should be able to pick SLOs/profiles according to their needs and a benchmark should enable them to do so. In benchmarking, it is also common to group applications together, for example, into latency-sensitive (L-apps) and throughput-sensitive apps (T-apps) [29], [37], [57], [75]. L-apps have stringent requirements on tail latency, such as caches. T-apps are batch workloads with constraints on the total runtime or average throughput, such as AI training. Other works classify their workloads as latency-critical (LC-app) and best-effort (BE-app) apps [18],

Take-away message

1. Modern NVMe storage requires performance isolation.
   - Low overhead, fairness, priority utilization trade-offs, priority burst support
2. We introduce isol-bench, a performance isolation benchmark suite.
3. cgroups is Linux's state-of-the-practice way to control isolation.
   - I/O schedulers do not provide good isolation.
   - io.cost provides the best isolation.

Paper: https://atlarge-research.com/pdfs/2025-iiswc-cgroups.pdf
Code: https://github.com/atlarge-research/isol-bench

Supported by NWO, Graph Massivizer, and 6G FNS

Further reading from our team

1. Zebin Ren, Krijn Doekemeijer, Nick Tehrany, and Animesh Trivedi. *BFQ, Multiqueue Deadline, or Kyber? Performance Characterization of Linux Storage Schedulers in the NVMe Era*. In Proceedings of the 15th ACM/SPEC International Conference on Performance Engineering (ICPE'24). 2024
2. Zebin Ren, Krijn Doekemeijer, Nick Tehrany, and Animesh Trivedi. *Performance Characterization of Modern Storage Stacks: POSIX I/O, libaio, SPDK, and io_uring.* In Proceedings of the 3th Workshop on Challenges and Opportunities of Efficient and Performant Storage Systems (CHEOPS'24). 2023.
3. Krijn Doekemeijer, Nick Tehrany, Balakrishnan Chandrasekaran, Matias Bjørling, and Animesh Trivedi. *Performance Characterization of NVMe Flash Devices with Zoned Namespaces (ZNS)*. 2023 IEEE International Conference on Cluster Computing (CLUSTER). 2023.
4. Animesh Trivedi, Matthijs Jansen, Krijn Doekemeijer, Sacheendra Talluri, and Nick Tehrany. *Reviving Storage Systems Education in the 21st Century — An experience report*. In 24th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGrid). 2024
5. Zebin Ren, Krijn Doekemeijer, Padma Apparao, and Animesh Trivedi. *Storage-Based Approximate Nearest Neighbor Search: What are the Performance Cost and I/O Characteristics?.* In 2025 IEEE International Symposium on Workload Characterization (IISWC). 2025.

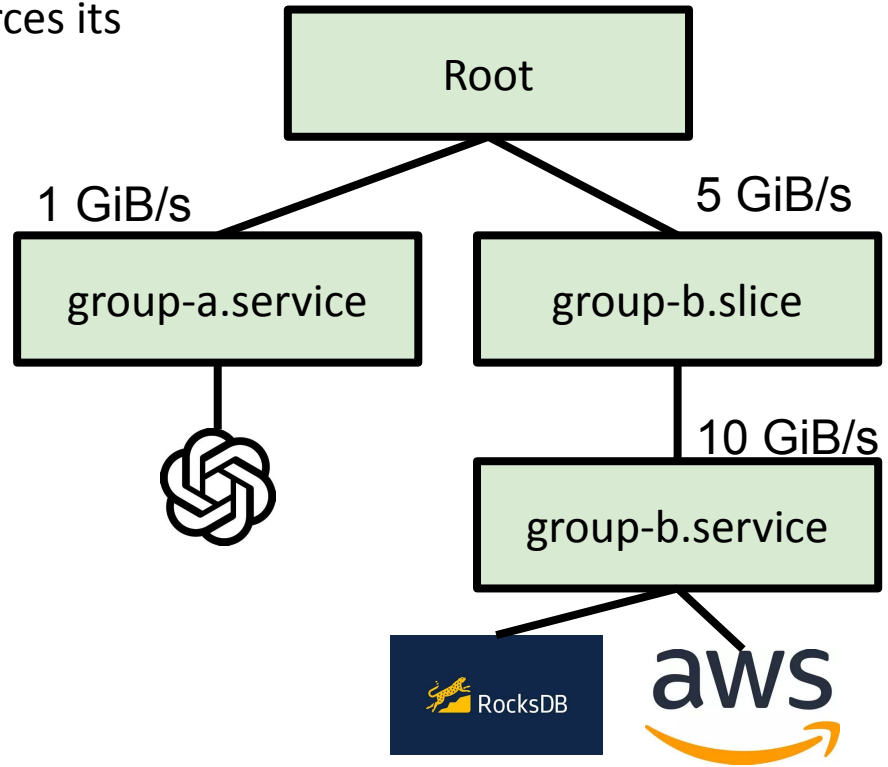
Backup slides

What to do next?

- Isolation for higher layers: file systems, caches
- Application-level isolation
- Combine knobs: e.g., io.cost + io.max

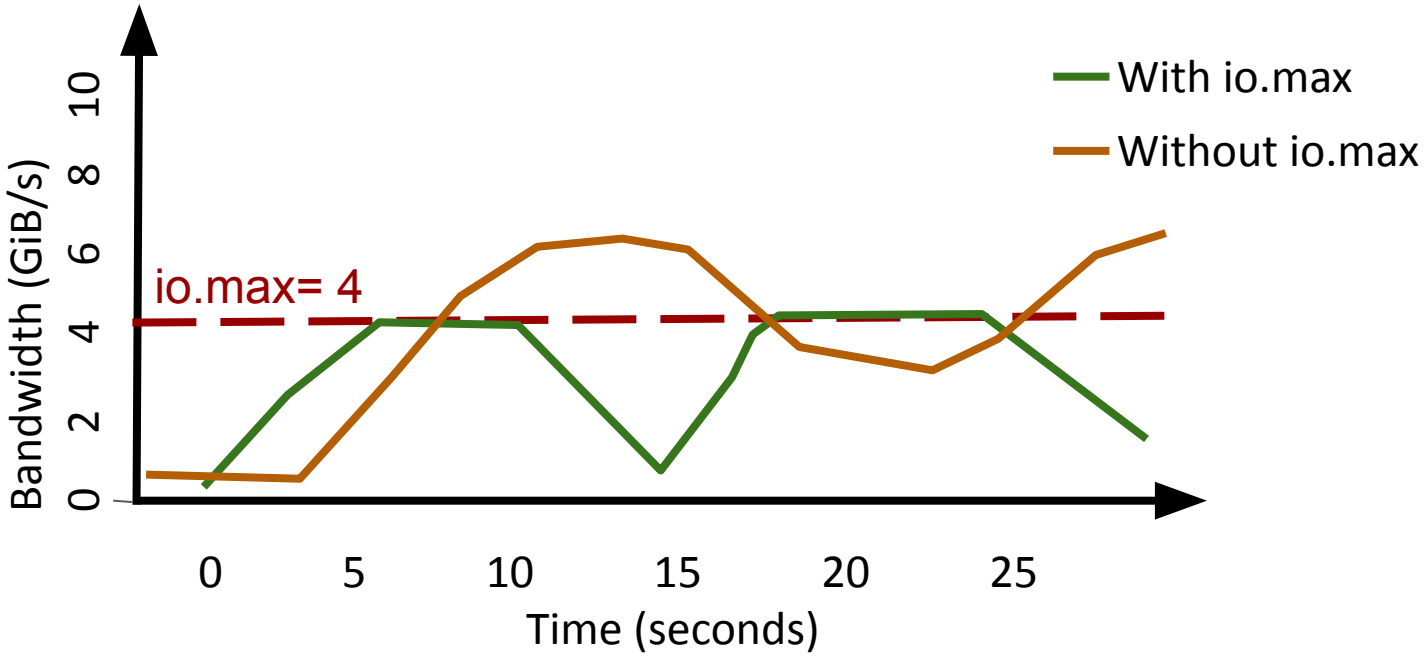
Background: cgroups

- In Linux all processes are part of a cgroup
- A cgroup, or control group, controls the resources its collective processes can use
- I/O control knobs limit storage resources
- cgroups is hierarchical
- In total 5 knobs:
  1. io.prio.class
     a. Requires mq-deadline scheduler
  2. Io.bfq.weight
     a. Requires BFQ scheduler
  3. **io.max**
  4. io.latency
  5. io.cost
- Schedulers have high overhead



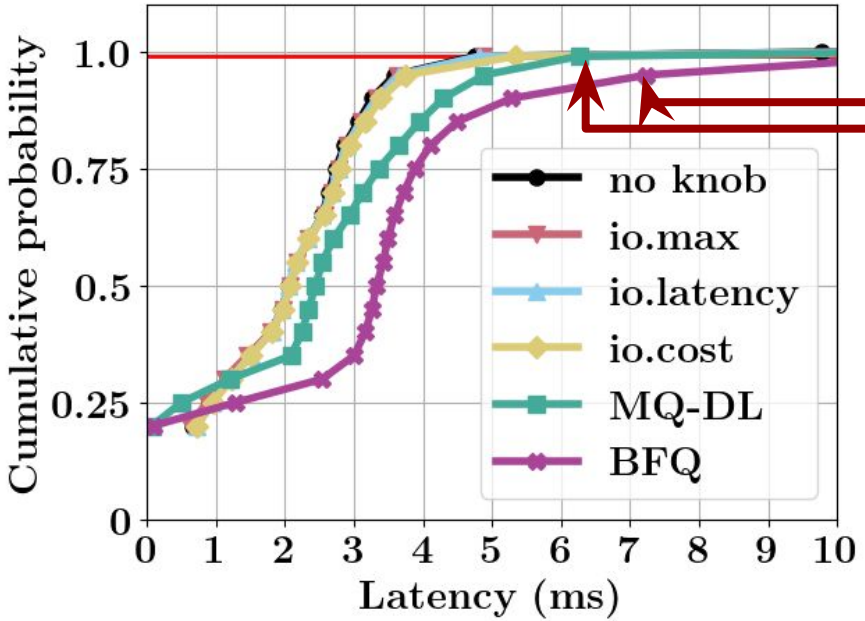
Example of a cgroups I/O knob: io.max

io.max allows setting max bandwidth (bytes) or throughput (operations) per cgroup

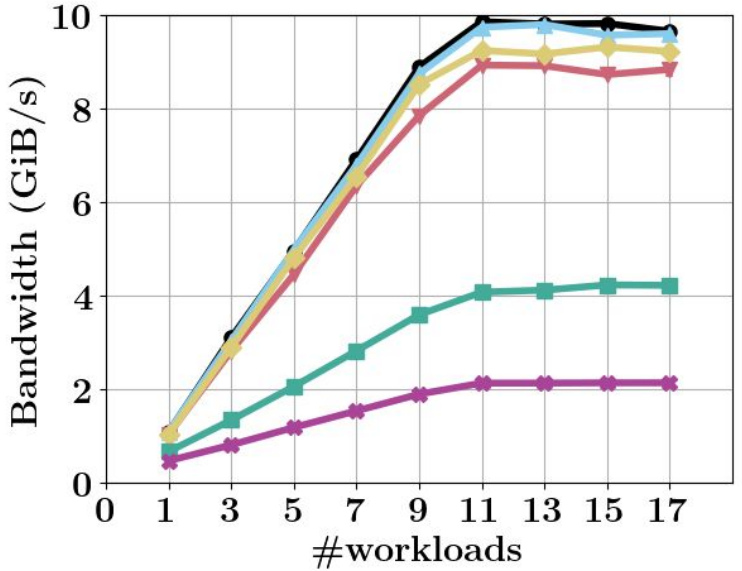


Desiderata 1: Low performance overhead

Latency CDF
(more to left is better)



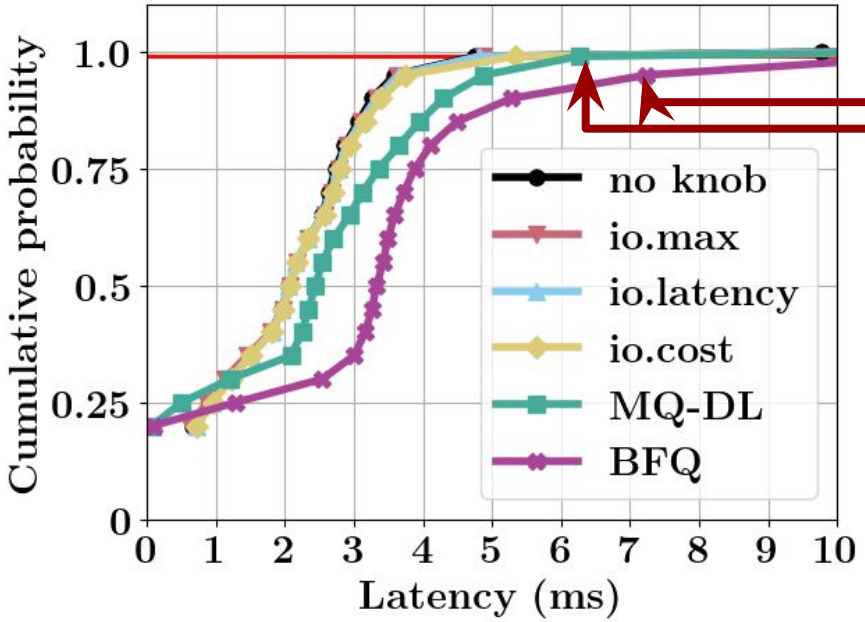
Bandwidth scalability with 7 SSDs
(higher is better)



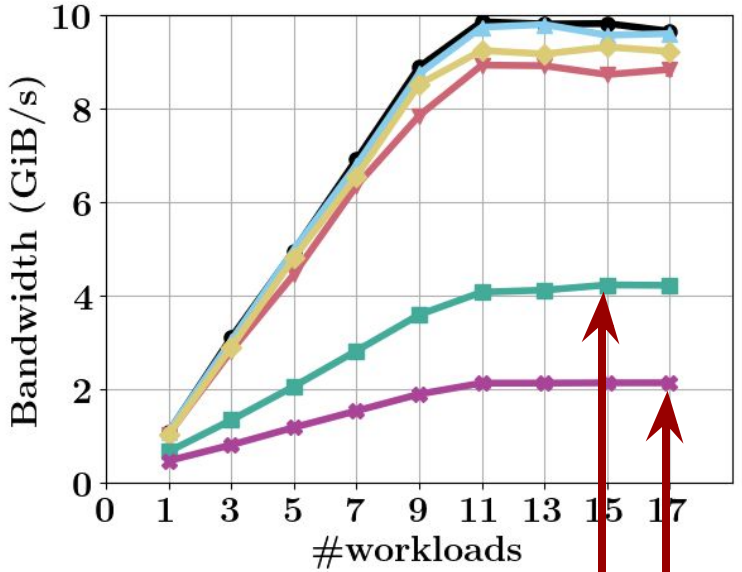
Schedulers have high overhead!

Desiderata 1: Low performance overhead

Latency CDF
(more to left is better)



Bandwidth scalability with 7 SSDs
(higher is better)



Schedulers have high overhead!