@Large Research
Massivizing Computer Systems

VU VRIJE UNIVERSITEIT AMSTERDAM

Characterizing Energy Efficiency Trade-offs in the Linux Storage Stack for Flash-based NVMe SSDs

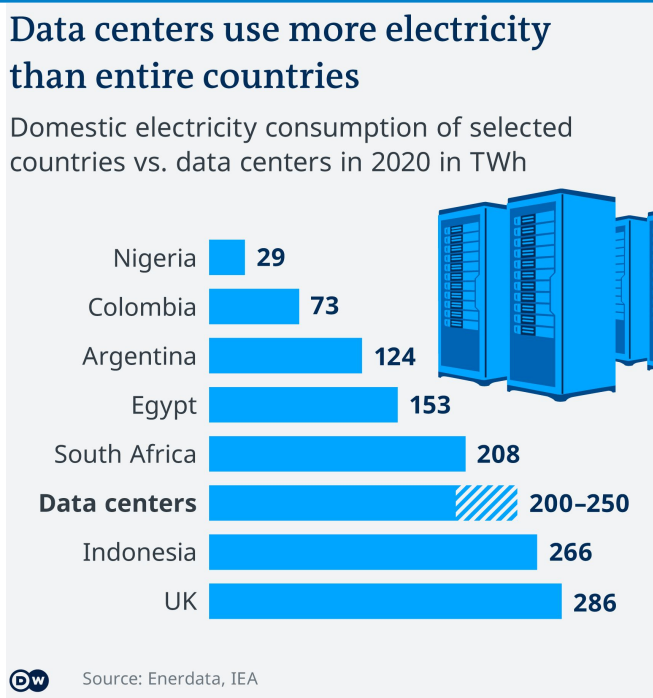
Joseph Kanichai*, Krijn Doekemeijer*, Steven van der Vlugt, Dante Niewenhuis, Tiziano De Matteis, and Animesh Trivedi

github.com/t348575

Data center (DC) energy is skyrocketing

How much energy?

- 1 DC => can easily reach 50–100MW
  - Close to Boulder's total consumption!
- Combined DCs => 2–6% of worldwide consumption, more than whole countries!



1 https://bouldercolorado.gov/boulder-xcel-energy-partnership-agreement-progress

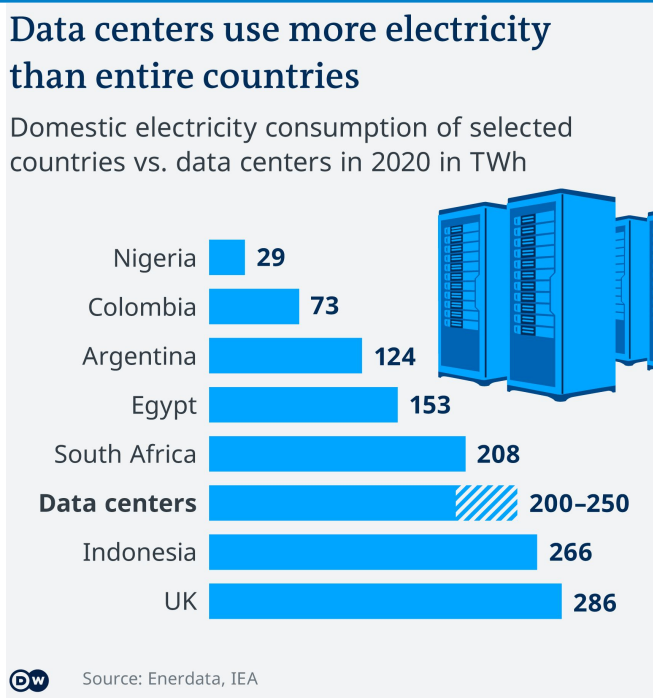
Data center (DC) energy is skyrocketing

How much energy?

- 1 DC => can easily reach 50–100MW
  - Close to Boulder's total consumption!
- Combined DCs => 2–6% of worldwide consumption, more than whole countries!

This work is part of a concerted effort to reduce storage energy use

- Storage is estimated to be >5% of DC energy

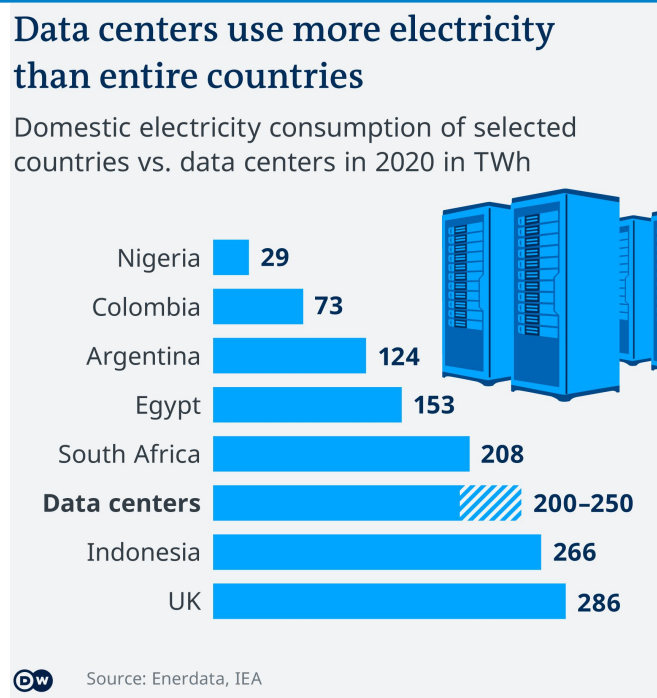


1 https://bouldercolorado.gov/boulder-xcel-energy-partnership-agreement-progress
2 www.iea.org/reports/energy-and-ai/energy-demand-from-ai, https://eta.lbl.gov/publications/2024-lbnl-data-center-energy-usage-report

Data center (DC) energy is skyrocketing

How much energy?

- 1 DC => can easily reach 50–100MW
  - Close to Boulder's total consumption!
- Combined DCs => 2–6% of worldwide consumption, more than whole countries!



This work is part of a concerted effort to reduce storage energy use

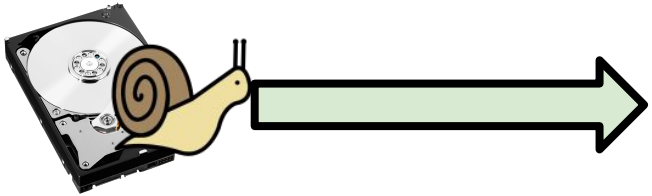
- Storage is estimated to be >5% of DC energy
- **Specifically, making <u>fast storage</u> energy efficient through software configurations.**

1 https://bouldercolorado.gov/boulder-xcel-energy-partnership-agreement-progress
2 www.iea.org/reports/energy-and-ai/energy-demand-from-ai, https://eta.lbl.gov/publications/2024-lbnl-data-center-energy-usage-report

Fast storage? DCs typically use **flash NVMe SSDs**

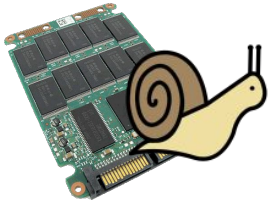
NVMe SSDs are significantly faster than HDDs AND SATA SSDs

HDD



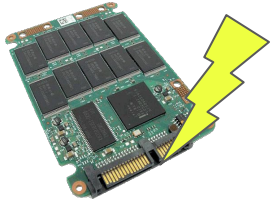
Latency: ms
Bandwidth: MiB/s

SATA SSD



Latency: 100s us
Bandwidth: MiB/s

NVMe SSD

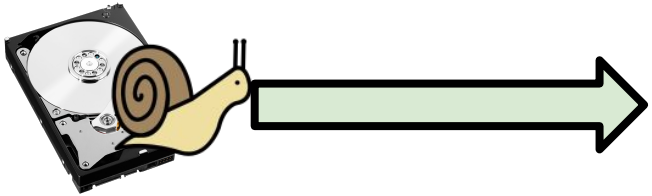


Latency: 10s us
Bandwidth: GiB/s

Fast storage? DCs typically use **flash NVMe SSDs**

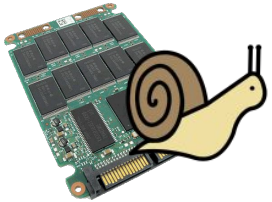
NVMe SSDs are significantly faster than HDDs AND SATA SSDs

HDD



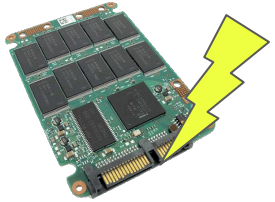
Latency: ms
Bandwidth: MiB/s

SATA SSD



Latency: 100s us
Bandwidth: MiB/s

NVMe SSD



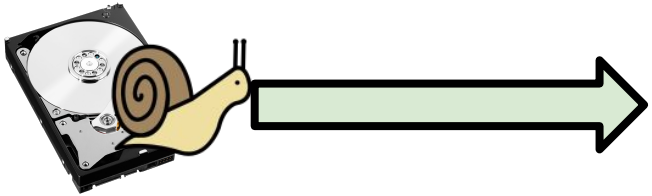
Latency: 10s us
Bandwidth: GiB/s

Software on the CPU is the bottleneck for NVMe SSD performance…

Fast storage? DCs typically use **flash NVMe SSDs**

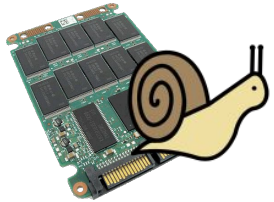
NVMe SSDs are significantly faster than HDDs AND SATA SSDs

HDD



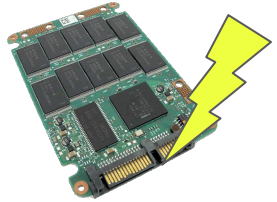
Latency: ms
Bandwidth: MiB/s

SATA SSD



Latency: 100s us
Bandwidth: MiB/s

NVMe SSD



Latency: 10s us
Bandwidth: GiB/s

Software on the CPU is the bottleneck for NVMe SSDs performance… =>
Now, we need to consider SSD *AND* CPU software for performance. How do we make workloads on such storage *energy efficient*?

VU

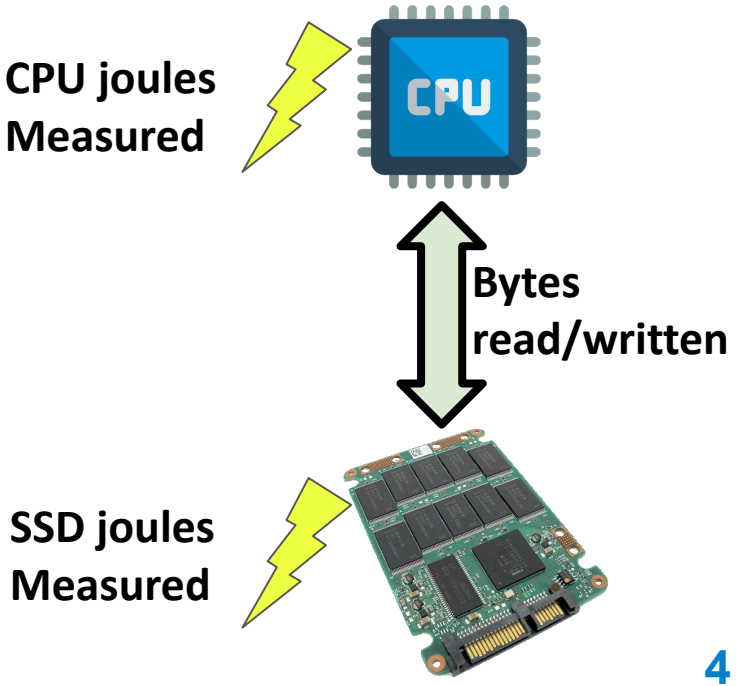
Key problem addressed in this work

“What are the energy efficiency trade-offs for workloads running on flash NVMe SSDs in Linux?”

Background: what is energy efficiency?

Energy efficiency is “the amount of work completed per joule”. We use:

Energy efficiency = (operations/bytes completed on the SSD) / joules



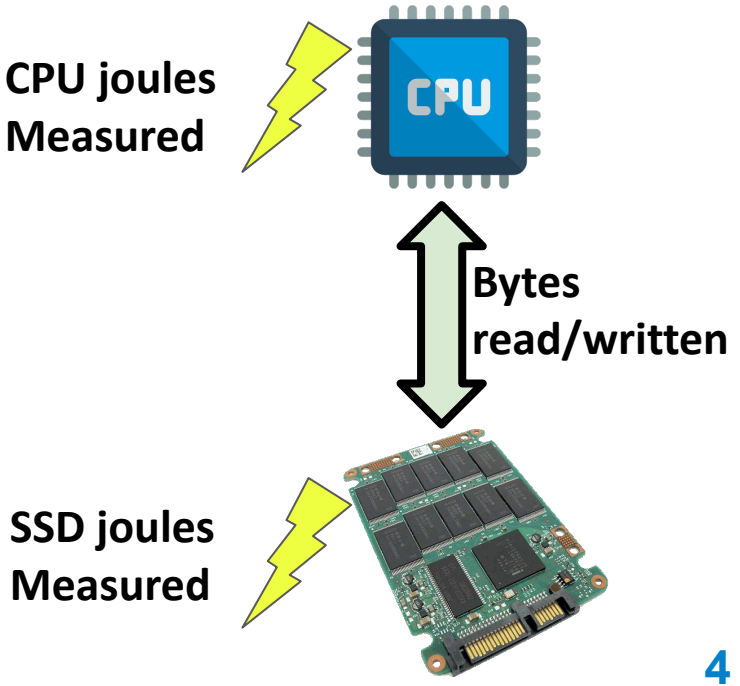
Background: what is energy efficiency?

Energy efficiency is “the amount of work completed per joule”. We use:

Energy efficiency = (operations/bytes completed on the SSD) / joules

Efficiency depends on the hardware measured:

1. SSD joules => SSD efficiency
2. CPU joules => CPU efficiency
3. Server joules => Server efficiency



So how do we “start” optimizing workloads on NVMe for efficiency?

Has this not been done before…

Research into storage energy efficiency itself is not novel; most of the pieces are there. However, when **we tried optimizing** a workload, we found an **end-to-end holistic understanding** to be missing:

Has this not been done before…

Research into storage energy efficiency itself is not novel; most of the pieces are there. However, when **we tried optimizing** a workload, we found an **end-to-end holistic understanding** to be missing:

1. **HW-specific.** Most studies limit to SATA OR selected NVMes
   - ➢ There is a recommendation difference between hardware…

Has this not been done before…

Research into storage energy efficiency itself is not novel; most of the pieces are there. However, when **we tried optimizing** a workload, we found an **end-to-end holistic understanding** to be missing:

1. **HW-specific.** Most studies limit to SATA OR selected NVMes
   - ➢ There is a recommendation difference between hardware…
2. **Attribution.** Most studies measure one component only…
   - ➢ Hides component (in)efficiencies, e.g., SSD, CPU, server…

Has this not been done before…

Research into storage energy efficiency itself is not novel; most of the pieces are there. However, when **we tried optimizing** a workload, we found an **end-to-end holistic understanding** to be missing:

1. **HW-specific.** Most studies limit to SATA OR selected NVMes
   - ➢ There is a recommendation difference between hardware…
2. **Attribution.** Most studies measure one component only…
   - ➢ Hides component (in)efficiencies, e.g., SSD, CPU, server…
3. **Complexity.** There are MANY system knobs (e.g., DVFS, power states, cgroups, …) to consider, and their interaction is understudied.
   - ➢ Which knobs matter in practice?

Has this not been done before…

Research into storage energy efficiency itself is not novel; most of the pieces are there. However, when **we tried optimizing** a workload, we found an **end-to-end holistic understanding** to be missing:

1. **HW-specific.** Most studies limit to SATA OR selected NVMes
   - ➢ There is a recommendation difference between hardware…
2. **Attribution.** Most studies measure one component only…
   - ➢ Hides component (in)efficiencies, e.g., SSD, CPU, server…
3. **Complexity.** There are MANY system knobs (e.g., DVFS, power states, cgroups, …) to consider, and their interaction is understudied.
   - ➢ Which knobs matter in practice?
4. **Methodology.** Lack of methodology for NVMe energy benchmarking
   - ➢ If behavior is setup-specific, how do we quickly test our own setup?

VU

What we ***can*** do to improve our understanding

1. A **benchmark** and/or methodology to repeat trade-off optima per setup
2. A **characterization** per hardware component in the Linux' stack for NVMe flash
3. An **understanding** of what configurations affect efficiency in Linux' for NVMe

What we ***did*** to improve our understanding

1. A **benchmark** and/or methodology to repeat trade-off optima per setup

   ➢ **nvme-energy-bench**, a modular, benchmark orchestrator

2. A **characterization** per hardware component in the Linux' stack for NVMe flash

   ➢ We **characterize** trade-offs, leading to seven generalizable observations

3. An **understanding** of what configurations affect efficiency in Linux' for NVMe

   ➢ In our **characterization,** we define what knobs to consider and how

The contributions we will discuss

1. nvme-energy-bench, a benchmark orchestrator

2. Efficiency characterization

Contribution 1 ⟶ nvme-energy-bench

We designed and implemented nvme-energy-bench, a bench orchestrator. It needed:

1. **Plug-n-play**
2. **Attribution per hardware component**
3. **Modular:**
    a. Sensors
    b. Configuration knobs
    c. Workloads (add your own!)
4. **Offline analysis**

Contribution 1 ⟶ nvme-energy-bench

We designed and implemented nvme-energy-bench, a bench orchestrator. It needed:

1. **Plug-n-play**
2. **Attribution per hardware component**
3. **Modular:**
    a. Sensors
    b. Configuration knobs
    c. Workloads (add your own!)
        i. No tie into one benchmark (suite)
4. **Offline analysis**

Contribution 1 ⟶ nvme-energy-bench

We designed and implemented nvme-energy-bench, a bench orchestrator. It needed:

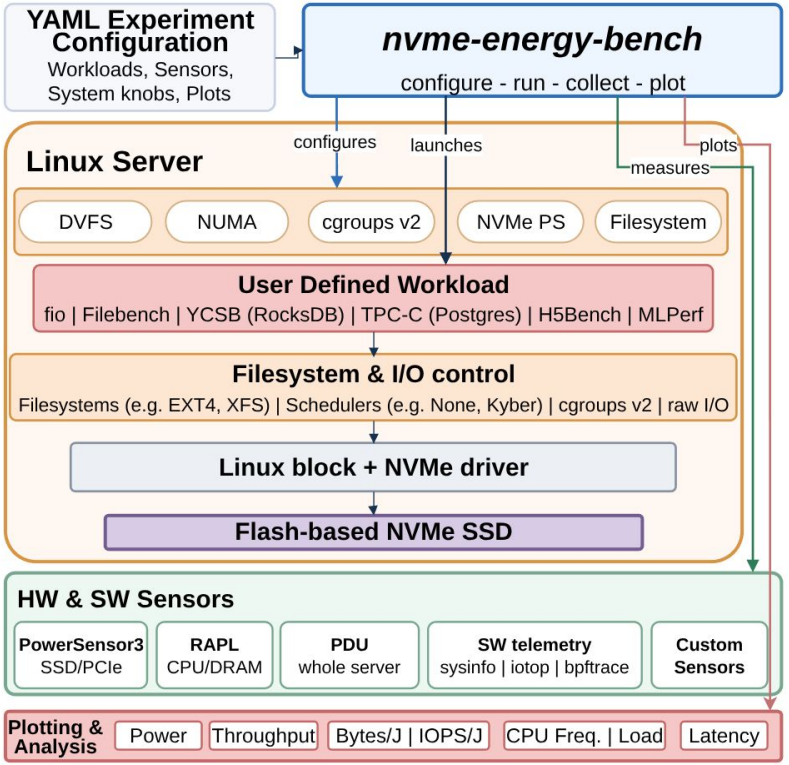
1. **Plug-n-play**
2. **Attribution per hardware component**
3. **Modular:**
    a. Sensors
    b. Configuration knobs
    c. Workloads (add your own!)
        i. No tie into one benchmark (suite)
4. **Offline analysis**

We will give a high-level top-down overview

nvme-energy-bench – orchestrator overview

A benchmark orchestrator that:

1. Sets power sensors
2. Sets global system knobs
3. Starts custom workloads
   a. Collects and stores metrics
4. Does offline analysis



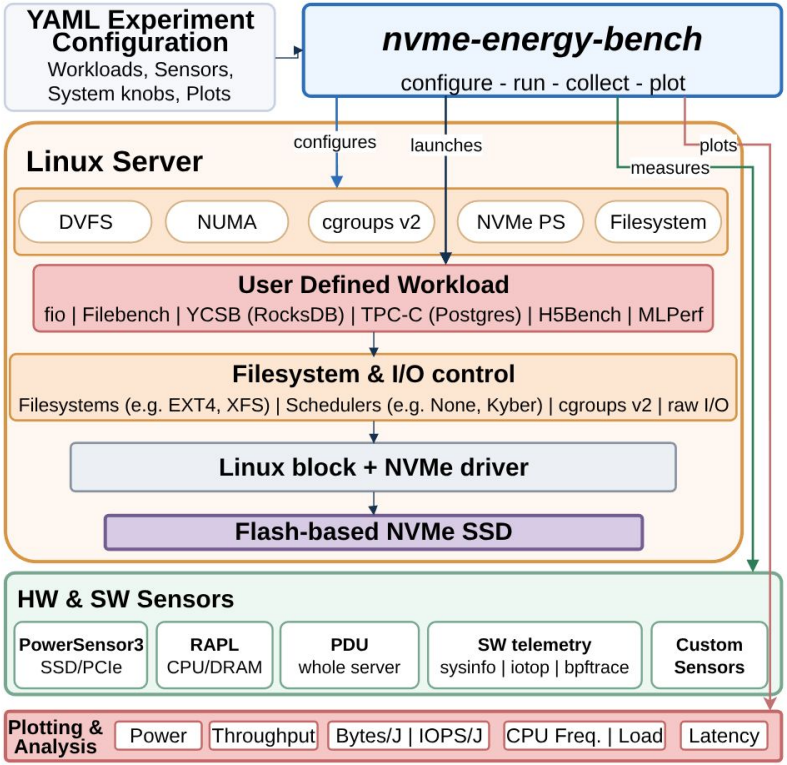
nvme-energy-bench – orchestrator overview

A benchmark orchestrator that:

1. Sets power sensors
2. Sets global system knobs
3. Starts custom workloads
    a. Collects and stores metrics
4. Does offline analysis

We will explain 1,2,3 in more detail.

Open-sourced!



nvme-energy-bench [1/3] – sensors

We *needed* support for:

1. Multiple independent HW & SW sensors
2. Custom sensors on availability
    a. Say you have custom sensors or hw

nvme-energy-bench [1/3] – sensors

We *added* support for:

1. Multiple independent HW & SW sensors
2. Custom sensors on availability

Add custom sensors with Rust interface implementation; then config sensors with YAML

```yaml
sensors:
  - sensor: Powersensor3
    args:
      type: Powersensor3Config
      device: /dev/ttyACM0
  - sensor: Rapl
  - sensor: Sysinfo
    args:
      type: SysinfoConfig
      interval: 10
  - sensor: NetioHttp
    args:
      type: NetioHttpConfig
      pdus:
        - alias: left
          url: http://192.168.1.78/netio.json
          loads: [Node3L]
        - alias: right
          url: http://192.168.1.79/netio.json
          loads: [Node3R]
  - sensor: Diskstat
```

nvme-energy-bench [1/3] – sensors

We *added* support for:

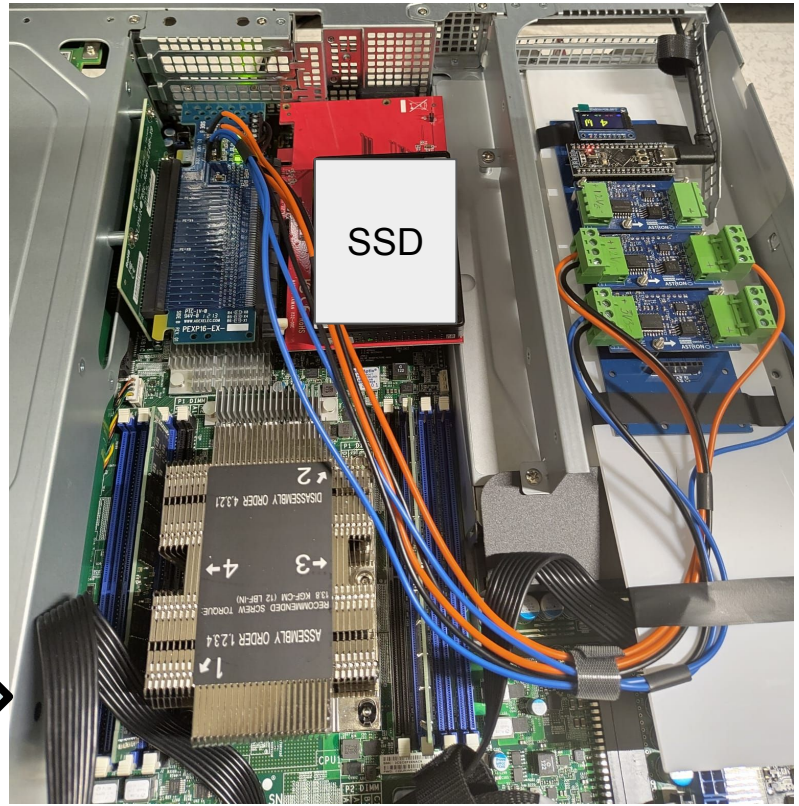
1. Multiple independent HW & SW sensors
2. Custom sensors on availability

Add custom sensors with Rust interface implementation; then config sensors with YAML

In our setup, we *used*:

1. Server -> 2x NetIO PowerPDU
2. CPU -> Software RAPL
3. SSD -> PowerSensor3

Visualization of our PowerSensor3 setup:



nvme-energy-bench [2/3] – knobs

We *needed* several knobs (**all covered partially!** in the literature):

1. Global
    a. I/O interfaces, including polling vs interrupts
    b. I/O control
    c. DVFS for CPU
    d. NVMe Power States (PS) for NVMe SSDs

nvme-energy-bench [2/3] – knobs

We *needed* several knobs (**all covered partially!** in the literature):

1. Global
    a. I/O interfaces, including polling vs interrupts
    b. I/O control
    c. DVFS for CPU
    d. NVMe Power States (PS) for NVMe SSDs
2. Workload (if applicable):
    a. I/O shaping
    b. Access patterns

nvme-energy-bench [3/3] – workloads

Additionally, we *needed* support for custom workloads:

1. KV stores
2. ML workloads
3. more…

nvme-energy-bench [3/3] – workloads

Additionally, we *needed* support for custom workloads:

1. KV stores
2. ML workloads
3. more…

Key is that we should be able to add workloads to not limit our orchestrator

nvme-energy-bench [3/3] – knobs + workloads

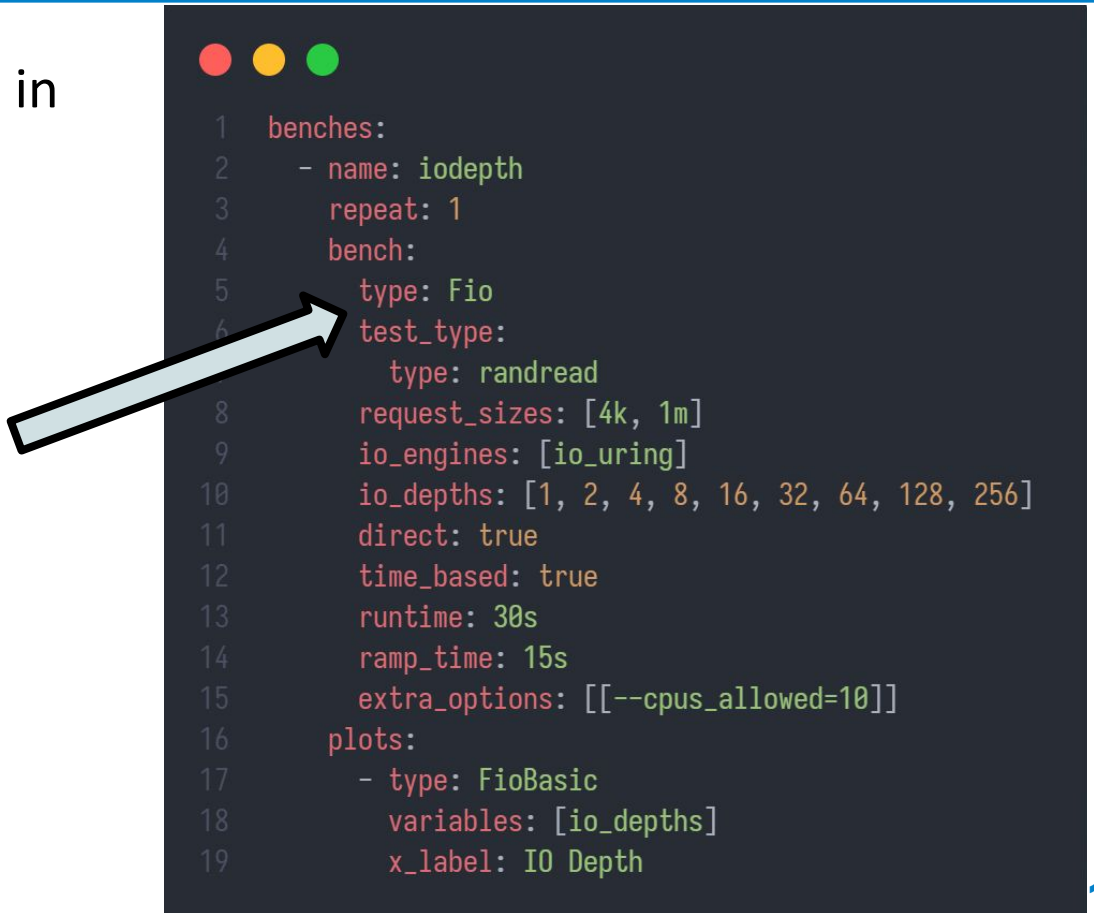
Workloads + knobs are defined in YAML:

1. Global knobs (e.g, DVFS)

nvme-energy-bench [3/3] – knobs + workloads

Workloads + knobs are defined in YAML:

1. Global knobs (e.g, DVFS)
2. Workloads, e.g.,
   a. **Fio micro-benchmarks**
   b. **KV**: YCSB
   c. **Custom**



```yaml
benches:
  - name: iodepth
    repeat: 1
    bench:
      type: Fio
      test_type:
        type: randread
      request_sizes: [4k, 1m]
      io_engines: [io_uring]
      io_depths: [1, 2, 4, 8, 16, 32, 64, 128, 256]
      direct: true
      time_based: true
      runtime: 30s
      ramp_time: 15s
      extra_options: [[--cpus_allowed=10]]
    plots:
      - type: FioBasic
        variables: [io_depths]
        x_label: IO Depth
```

In-between recap

RQ: “What are the energy efficiency trade-offs for workloads running on flash NVMe SSDs in Linux?”

Our contributions:

1. [discussed] nvme-energy-bench, a benchmark orchestrator
2. [to discuss] efficiency characterization

Contribution 2 ⟶ Efficiency characterization

We characterized on the following CPU-centric hardware setup

- **Server:** Supermicro SYS-2029GP-TR
- **SSD:** 5 NVMe SSD models from multiple vendors
- **CPU:** Intel Xeon Silver 4210 20 cores
- **DRAM:** 256 GB DDR4

Contribution 2 ⟶ Efficiency characterization

We characterized on the following CPU-centric hardware setup

- **Server:** Supermicro SYS-2029GP-TR
- **SSD:** 5 NVMe SSD models from multiple vendors
- **CPU:** Intel Xeon Silver 4210 20 cores
- **DRAM:** 256 GB DDR4

We orchestrated the following benchmarks:

- Micro-benchmark: *fio*
- Application workloads: *YCSB, varmail, fileserver*

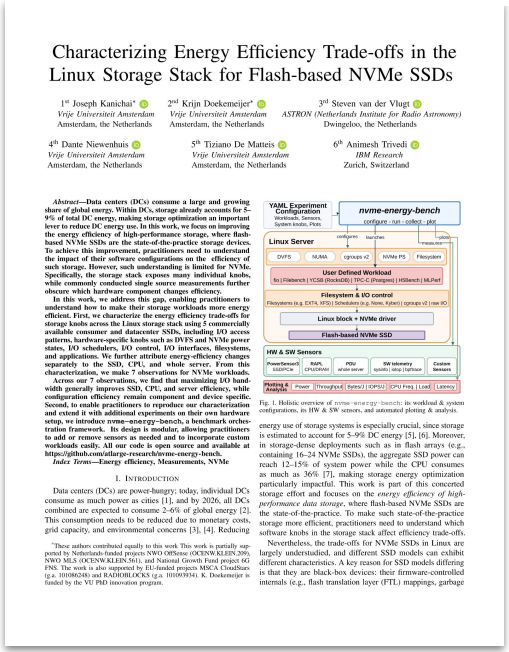
We did a sweep of *all* knobs (e.g., access patterns, I/O schedulers)

Contribution 2 → Efficiency characterization

See our paper for all results!

We will discuss *three* of our results, demonstrating nvme-energy-bench's use:

7 observations!



Characterizing Energy Efficiency Trade-offs in the Linux Storage Stack for Flash-based NVMe SSDs

1st Joseph Kanichai* — *Vrije Universiteit Amsterdam*, Amsterdam, the Netherlands
2nd Krijn Doekemeijer* — *Vrije Universiteit Amsterdam*, Amsterdam, the Netherlands
3rd Steven van der Vlugt — *ASTRON (Netherlands Institute for Radio Astronomy)*, Dwingeloo, the Netherlands
4th Dante Niewenhuis — *Vrije Universiteit Amsterdam*, Amsterdam, the Netherlands
5th Tiziano De Matteis — *Vrije Universiteit Amsterdam*, Amsterdam, the Netherlands
6th Animesh Trivedi — *IBM Research*, Zurich, Switzerland

***Abstract*—Data centers (DCs) consume a large and growing share of global energy. Within DCs, storage already accounts for 5–9% of total DC energy, making storage optimization an important lever to reduce DC energy use. In this work, we focus on improving the energy efficiency of high-performance storage, where flash-based NVMe SSDs are the state-of-the-practice storage devices. To achieve this improvement, practitioners need to understand the impact of their software configurations on the efficiency of such storage. However, such understanding is limited for NVMe. Specifically, the storage stack exposes many individual knobs, while commonly conducted single source measurements further obscure which hardware component changes efficiency.**

In this work, we address this gap, enabling practitioners to understand how to make their storage workloads more energy efficient. First, we characterize the energy efficiency trade-offs for storage knobs across the Linux storage stack using 5 commercially available consumer and datacenter SSDs, including I/O access patterns, hardware-specific knobs such as DVFS and NVMe power states, I/O schedulers, I/O control, I/O interfaces, filesystems, and applications. We further attribute energy-efficiency changes separately to the SSD, CPU, and whole server. From this characterization, we make 7 observations for NVMe workloads.

Across our 7 observations, we find that maximizing I/O bandwidth generally improves SSD, CPU, and server efficiency, while configuration efficiency remain component and device specific. Second, to enable practitioners to reproduce our characterization and extend it with additional experiments on their own hardware setup, we introduce nvme-energy-bench, a benchmark orchestration framework. Its design is modular, allowing practitioners to add or remove sensors as needed and to incorporate custom workloads easily. All our code is open source and available at https://github.com/atlarge-research/nvme-energy-bench.

***Index Terms*—Energy efficiency, Measurements, NVMe**

I. Introduction

Data centers (DCs) are power-hungry; today, individual DCs consume as much power as cities [1], and by 2026, all DCs combined are expected to consume 2–6% of global energy [2]. This consumption needs to be reduced due to monetary costs, grid capacity, and environmental concerns [3], [4]. Reducing energy use of storage systems is especially crucial, since storage is estimated to account for 5–9% DC energy [5], [6]. Moreover, in storage-dense deployments such as in flash arrays (e.g., containing 16–24 NVMe SSDs), the aggregate SSD power can reach 12–15% of system power while the CPU consumes as much as 36% [7], making storage energy optimization particularly impactful. This work is part of this concerted storage effort and focuses on the *energy efficiency of high-performance data storage*, where flash-based NVMe SSDs are the state-of-the-practice. To make such state-of-the-practice storage more efficient, practitioners need to understand which software knobs in the storage stack affect efficiency trade-offs.

Nevertheless, the trade-offs for NVMe SSDs in Linux are largely understudied, and different SSD models can exhibit different characteristics. A key reason for SSD models differing is that they are black-box devices: their firmware-controlled internals (e.g., flash translation layer (FTL) mappings, garbage

*These authors contributed equally to this work This work is partially supported by Netherlands-funded projects NWO OffSense (OCENWKLEIN.209), NWO MLS (OCENWKLEIN.561), and National Growth Fund project 6G INS. The work is also supported by EU-funded projects MSCA CloudStars (g.a. 101086248) and RADIOBLOCKS (g.a. 101093934). K. Doekemeijer is funded by the VU PhD innovation program.

Fig. 1. Holistic overview of nvme-energy-bench: its workload & system configurations, its HW & SW sensors, and automated plotting & analysis.

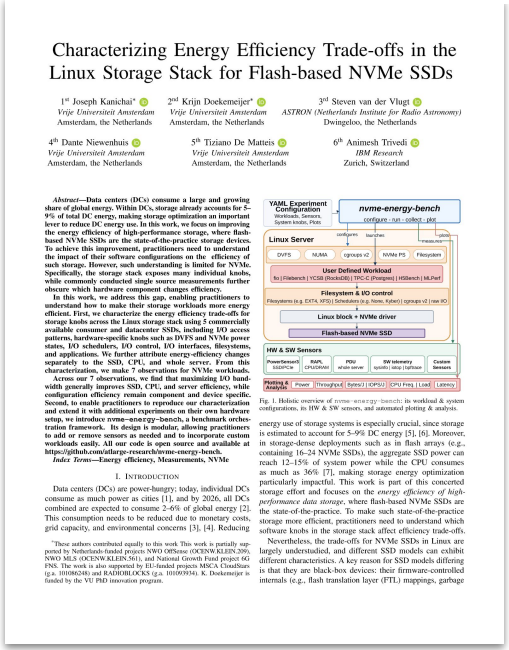
Contribution 2 → Efficiency characterization

See our paper for all results!

We will discuss *three* of our results, demonstrating nvme-energy-bench's use:

7 observations!

1. **Bandwidth**
   - ➢ BW is most important for efficiency.



Characterizing Energy Efficiency Trade-offs in the Linux Storage Stack for Flash-based NVMe SSDs

1st Joseph Kanichai* — Vrije Universiteit Amsterdam, Amsterdam, the Netherlands
2nd Krijn Doekemeijer* — Vrije Universiteit Amsterdam, Amsterdam, the Netherlands
3rd Steven van der Vlugt — ASTRON (Netherlands Institute for Radio Astronomy), Dwingeloo, the Netherlands
4th Dante Niewenhuis — Vrije Universiteit Amsterdam, Amsterdam, the Netherlands
5th Tiziano De Matteis — Vrije Universiteit Amsterdam, Amsterdam, the Netherlands
6th Animesh Trivedi — IBM Research, Zurich, Switzerland

***Abstract*—Data centers (DCs) consume a large and growing share of global energy. Within DCs, storage already accounts for 5–9% of total DC energy, making storage optimization an important lever to reduce DC energy use. In this work, we focus on improving the energy efficiency of high-performance storage, where flash-based NVMe SSDs are the state-of-the-practice storage devices. To achieve this improvement, practitioners need to understand the impact of their software configurations on the efficiency of such storage. However, such understanding is limited for NVMe. Specifically, the storage stack exposes many individual knobs, while commonly conducted single source measurements further obscure which hardware component changes efficiency.**

In this work, we address this gap, enabling practitioners to understand how to make their storage workloads more energy efficient. First, we characterize the energy efficiency trade-offs for storage knobs across the Linux storage stack using 5 commercially available consumer and datacenter SSDs, including I/O access patterns, hardware-specific knobs such as DVFS and NVMe power states, I/O schedulers, I/O control, I/O interfaces, filesystems, and applications. We further attribute energy-efficiency changes separately to the SSD, CPU, and whole server. From this characterization, we make 7 observations for NVMe workloads.

Across our 7 observations, we find that maximizing I/O bandwidth generally improves SSD, CPU, and server efficiency, while configuration efficiency remain component and device specific. Second, to enable practitioners to reproduce our characterization and extend it with additional experiments on their own hardware setup, we introduce nvme-energy-bench, a benchmark orchestration framework. Its design is modular, allowing practitioners to add or remove sensors as needed and to incorporate custom workloads easily. All our code is open source and available at https://github.com/atlarge-research/nvme-energy-bench.

***Index Terms*—Energy efficiency, Measurements, NVMe**

I. Introduction

Data centers (DCs) are power-hungry; today, individual DCs consume as much power as cities [1], and by 2026, all DCs combined are expected to consume 2–6% of global energy [2]. This consumption needs to be reduced due to monetary costs, grid capacity, and environmental concerns [3], [4]. Reducing energy use of storage systems is especially crucial, since storage is estimated to account for 5–9% DC energy [5], [6]. Moreover, in storage-dense deployments such as in flash arrays (e.g., containing 16–24 NVMe SSDs), the aggregate SSD power can reach 12–15% of system power while the CPU consumes as much as 36% [7], making storage energy optimization particularly impactful. This work is part of this concerted storage effort and focuses on the *energy efficiency of high-performance data storage*, where flash-based NVMe SSDs are the state-of-the-practice. To make such state-of-the-practice storage more efficient, practitioners need to understand which software knobs in the storage stack affect efficiency trade-offs.

Nevertheless, the trade-offs for NVMe SSDs in Linux are largely understudied, and different SSD models can exhibit different characteristics. A key reason for SSD models differing is that they are black-box devices: their firmware-controlled internals (e.g., flash translation layer (FTL) mappings, garbage

*These authors contributed equally to this work This work is partially supported by Netherlands-funded projects NWO OffSense (OCENW.KLEIN.209), NWO MLS (OCENW.KLEIN.561), and National Growth Fund project 6G INS. The work is also supported by EU-funded projects MSCA CloudStars (g.a. 101086248) and RADIOBLOCKS (g.a. 101093934). K. Doekemeijer is funded by the VU PhD innovation program.

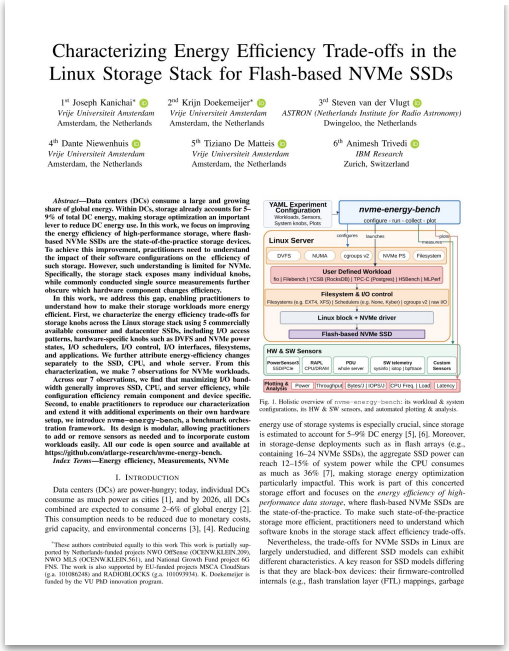
Contribution 2 → Efficiency characterization

See our paper for all results!

We will discuss *three* of our results, demonstrating nvme-energy-bench's use:

7 observations!

1. **Bandwidth**
   - ➢ BW is most important for efficiency.
2. **Knob overview**
   - ➢ Which knobs impact which hw component?
   - ➢ Does different hw need different knobs?



Characterizing Energy Efficiency Trade-offs in the Linux Storage Stack for Flash-based NVMe SSDs

1st Joseph Kanichai* — Vrije Universiteit Amsterdam, Amsterdam, the Netherlands
2nd Krijn Doekemeijer* — Vrije Universiteit Amsterdam, Amsterdam, the Netherlands
3rd Steven van der Vlugt — ASTRON (Netherlands Institute for Radio Astronomy), Dwingeloo, the Netherlands
4th Dante Niewenhuis — Vrije Universiteit Amsterdam, Amsterdam, the Netherlands
5th Tiziano De Matteis — Vrije Universiteit Amsterdam, Amsterdam, the Netherlands
6th Animesh Trivedi — IBM Research, Zurich, Switzerland

Abstract—Data centers (DCs) consume a large and growing share of global energy. Within DCs, storage already accounts for 5–9% of total DC energy, making storage optimization an important lever to reduce DC energy use. In this work, we focus on improving the energy efficiency of high-performance storage, where flash-based NVMe SSDs are the state-of-the-practice storage devices. To achieve this improvement, practitioners need to understand the impact of their software configurations on the efficiency of such storage. However, such understanding is limited for NVMe. Specifically, the storage stack exposes many individual knobs, while commonly conducted single source measurements further obscure which hardware component changes efficiency.

In this work, we address this gap, enabling practitioners to understand how to make their storage workloads more energy efficient. First, we characterize the energy efficiency trade-offs for storage knobs across the Linux storage stack using 5 commercially available consumer and datacenter SSDs, including I/O access patterns, hardware-specific knobs such as DVFS and NVMe power states, I/O schedulers, I/O control, I/O interfaces, filesystems, and applications. We further attribute energy-efficiency changes separately to the SSD, CPU, and whole server. From this characterization, we make 7 observations for NVMe workloads.

Across our 7 observations, we find that maximizing I/O bandwidth generally improves SSD, CPU, and server efficiency, while configuration efficiency remain component and device specific. Second, to enable practitioners to reproduce our characterization and extend it with additional experiments on their own hardware setup, we introduce nvme-energy-bench, a benchmark orchestration framework. Its design is modular, allowing practitioners to add or remove sensors as needed and to incorporate custom workloads easily. All our code is open source and available at https://github.com/atlarge-research/nvme-energy-bench.

Index Terms—Energy efficiency, Measurements, NVMe

I. Introduction

Data centers (DCs) are power-hungry; today, individual DCs consume as much power as cities [1], and by 2026, all DCs combined are expected to consume 2–6% of global energy [2]. This consumption needs to be reduced due to monetary costs, grid capacity, and environmental concerns [3], [4]. Reducing energy use of storage systems is especially crucial, since storage is estimated to account for 5–9% DC energy [5], [6]. Moreover, in storage-dense deployments such as in flash arrays (e.g., containing 16–24 NVMe SSDs), the aggregate SSD power can reach 12–15% of system power while the CPU consumes as much as 36% [7], making storage energy optimization particularly impactful. This work is part of this concerted storage effort and focuses on the *energy efficiency of high-performance data storage*, where flash-based NVMe SSDs are the state-of-the-practice. To make such state-of-the-practice storage more efficient, practitioners need to understand which software knobs in the storage stack affect efficiency trade-offs.

Nevertheless, the trade-offs for NVMe SSDs in Linux are largely understudied, and different SSD models can exhibit different characteristics. A key reason for SSD models differing is that they are black-box devices: their firmware-controlled internals (e.g., flash translation layer (FTL) mappings, garbage

*These authors contributed equally to this work This work is partially supported by Netherlands-funded projects NWO OffSense (OCENW.KLEIN.209), NWO MLS (OCENW.KLEIN.561), and National Growth Fund project 6G INS. The work is also supported by EU-funded projects MSCA CloudStars (g.a. 101086248) and RADIOBLOCKS (g.a. 101093934). K. Doekemeijer is funded by the VU PhD innovation program.

Contribution 2 → Efficiency characterization

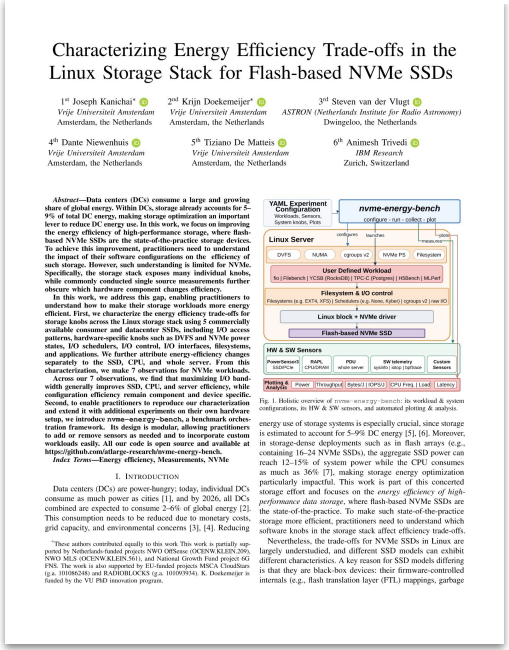
See our paper for all results!

We will discuss *three* of our results, demonstrating nvme-energy-bench's use:

7 observations!

1. **Bandwidth**
   - ➢ BW is most important for efficiency.
2. **Knob overview**
   - ➢ Which knobs impact which hw component?
   - ➢ Does different hw need different knobs?
3. **Attribution**
   - ➢ Why we need to measure all components

And a lot more in the paper…



Characterizing Energy Efficiency Trade-offs in the Linux Storage Stack for Flash-based NVMe SSDs

1st Joseph Kanichai* Vrije Universiteit Amsterdam Amsterdam, the Netherlands
2nd Krijn Doekemeijer* Vrije Universiteit Amsterdam Amsterdam, the Netherlands
3rd Steven van der Vlugt ASTRON (Netherlands Institute for Radio Astronomy) Dwingeloo, the Netherlands
4th Dante Niewenhuis Vrije Universiteit Amsterdam Amsterdam, the Netherlands
5th Tiziano De Matteis Vrije Universiteit Amsterdam Amsterdam, the Netherlands
6th Animesh Trivedi IBM Research Zurich, Switzerland

***Abstract*—Data centers (DCs) consume a large and growing share of global energy. Within DCs, storage already accounts for 5–9% of total DC energy, making storage optimization an important lever to reduce DC energy use. In this work, we focus on improving the energy efficiency of high-performance storage, where flash-based NVMe SSDs are the state-of-the-practice storage devices. To achieve this improvement, practitioners need to understand the impact of their software configurations on the efficiency of such storage. However, such understanding is limited for NVMe. Specifically, the storage stack exposes many individual knobs, while commonly conducted single source measurements further obscure which hardware component changes efficiency.**

In this work, we address this gap, enabling practitioners to understand how to make their storage workloads more energy efficient. First, we characterize the energy efficiency trade-offs for storage knobs across the Linux storage stack using 5 commercially available consumer and datacenter SSDs, including I/O access patterns, hardware-specific knobs such as DVFS and NVMe power states, I/O schedulers, I/O control, I/O interfaces, filesystems, and applications. We further attribute energy-efficiency changes separately to the SSD, CPU, and whole server. From this characterization, we make 7 observations for NVMe workloads.

Across our 7 observations, we find that maximizing I/O bandwidth generally improves SSD, CPU, and server efficiency, while configuration efficiency remain component and device specific. Second, to enable practitioners to reproduce our characterization and extend it with additional experiments on their own hardware setup, we introduce nvme-energy-bench, a benchmark orchestration framework. Its design is modular, allowing practitioners to add or remove sensors as needed and to incorporate custom workloads easily. All our code is open source and available at https://github.com/astron-rd/nvme-energy-bench.

***Index Terms*—Energy efficiency, Measurements, NVMe**

I. Introduction

Data centers (DCs) are power-hungry; today, individual DCs consume as much power as cities [1], and by 2026, all DCs combined are expected to consume 2–6% of global energy [2]. This consumption needs to be reduced due to monetary costs, grid capacity, and environmental concerns [3], [4]. Reducing energy use of storage systems is especially crucial, since storage is estimated to account for 5–9% DC energy [5], [6]. Moreover, in storage-dense deployments such as in flash arrays (e.g., containing 16–24 NVMe SSDs), the aggregate SSD power can reach 12–15% of system power while the CPU consumes as much as 36% [7], making storage energy optimization particularly impactful. This work is part of this concerted storage effort and focuses on the *energy efficiency of high-performance data storage*, where flash-based NVMe SSDs are the state-of-the-practice. To make such state-of-the-practice storage more efficient, practitioners need to understand which software knobs in the storage stack affect efficiency trade-offs.

Nevertheless, the trade-offs for NVMe SSDs in Linux are largely understudied, and different SSD models can exhibit different characteristics. A key reason for SSD models differing is that they are black-box devices: their firmware-controlled internals (e.g., flash translation layer (FTL) mappings, garbage

*These authors contributed equally to this work This work is partially supported by Netherlands-funded projects NWO OffSense (OCENWKLEIN.209), NWO MLS (OCENWKLEIN.561), and National Growth Fund project 6G INS. The work is also supported by EU-funded projects MSCA CloudStars (g.a. 101086248) and RADIOBLOCKS (g.a. 101093934). K. Doekemeijer is funded by the VU PhD innovation program.

Efficiency characterization [1/3] - Bandwidth

Increasing bandwidth can improve efficiency across the hw components.

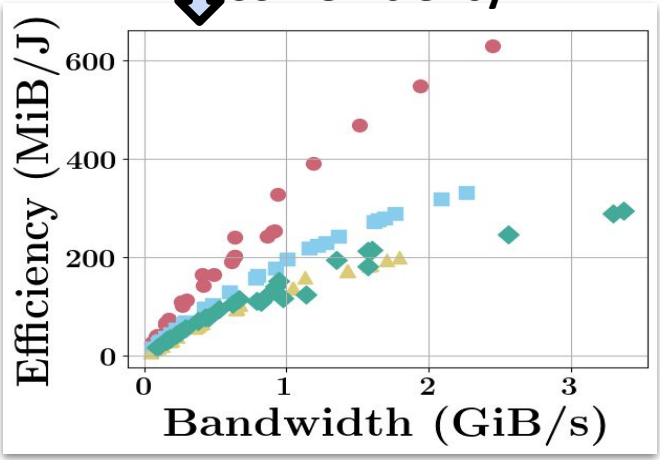
- Efficiency optimization through bandwidth optimization!
- We will give an example per component

Efficiency characterization [1/3] - Bandwidth

Increasing bandwidth can improve efficiency across the hw components:

1. For the four SSDs

SSD efficiency



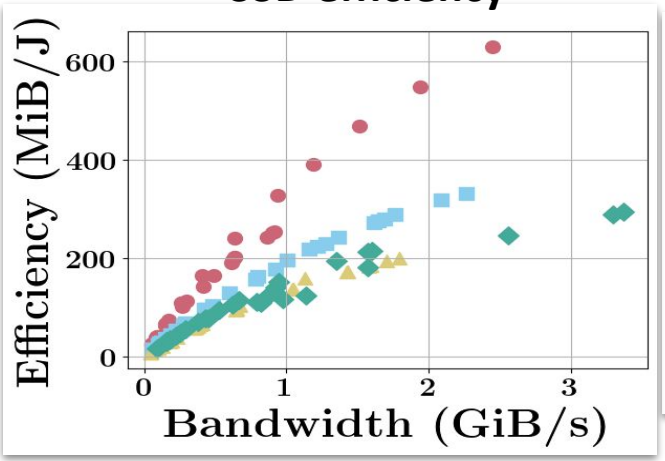
SSD model: A B C D

Efficiency characterization [1/3] - Bandwidth

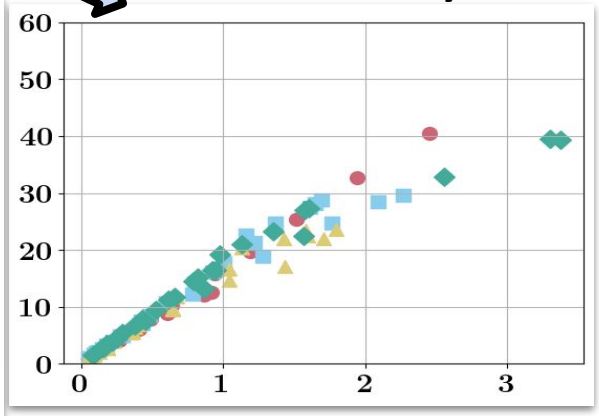
Increasing bandwidth can improve efficiency across the hw components:

1. For the four SSDs
2. For the CPU

SSD efficiency



CPU efficiency



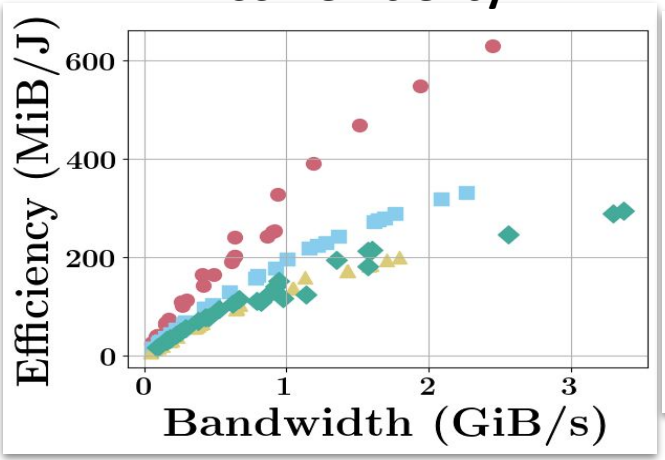
SSD model: A B C D

Efficiency characterization [1/3] - Bandwidth

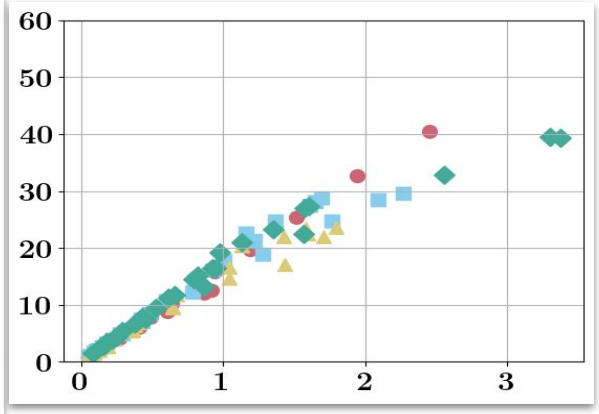
Increasing bandwidth can improve efficiency across the hw components:

1. For the four SSDs
2. For the CPU
3. For the server; the server follows CPU trends in our setup (CPU power dominates)

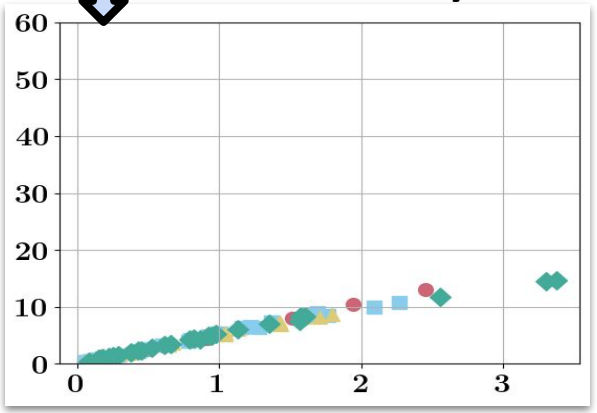
SSD efficiency



CPU efficiency



Server efficiency



SSD model: A B C D

Efficiency characterization [2/3] - Knob overview

Our knobs (from our FULL sweep) impact hardware differently:

- HW components require different knobs
- We give an example of our setup

Efficiency characterization [2/3] - Knob overview

Overview of knob impact on per component efficiency in our setup

Knob	SSD	CPU	Server
I/O shaping	●	●	●; Follows CPU
Access patterns	●	✗	✗ ; Follows CPU
I/O interface	✗	●	●; Follows CPU
DVFS	✗	●	●; Follows CPU
NVMe power states	◐	✗	✗ ; Follows CPU
I/O control	✗	◐	◐; Follows CPU

✗ = no impact ◐ = slight impact ● = impact

Efficiency characterization [2/3] - Knob overview

Overview of knob impact on per component efficiency in our setup

Knob	SSD	CPU	Server
I/O shaping	●	●	**●; Follows CPU**
Access patterns	●	✗	✗ ; Follows CPU
I/O interface	✗	●	●; Follows CPU
DVFS	✗	●	●; Follows CPU
NVMe power states	◐	✗	✗ ; Follows CPU
I/O control	✗	◐	◐; Follows CPU

✗ = no impact ◐ = slight impact ● = impact

Efficiency characterization [2/3] - Knob overview

Overview of knob impact on per component efficiency in our setup

Knob	SSD	CPU	Server
I/O shaping	●	●	●; Follows CPU
Access patterns	●	✗	**✗; Follows CPU**
I/O interface	**✗**	●	**●; Follows CPU**
DVFS	✗	●	●; Follows CPU
NVMe power states	◐	✗	✗; Follows CPU
I/O control	✗	◐	◐; Follows CPU

✗ = no impact ◐ = slight impact ● = impact

Efficiency characterization [3/3] - Attribution

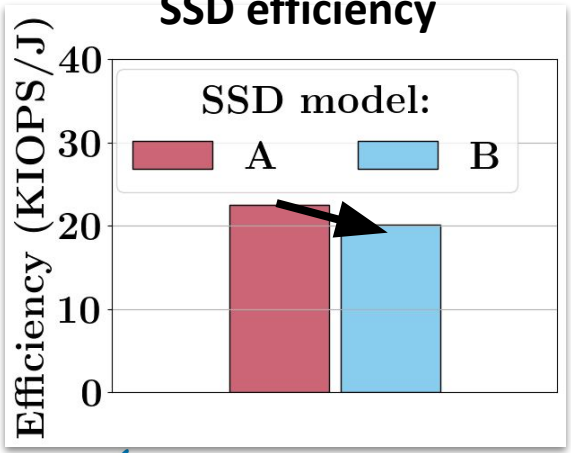
Dependent on the component measured, efficiency recommendations change.

We demonstrate this attribution problem by changing an SSD model.

Efficiency characterization [3/3] - Attribution

We change SSD from “A” to “B” for the varmail workload:

1. **If we look at the SSD** → A is more efficient.

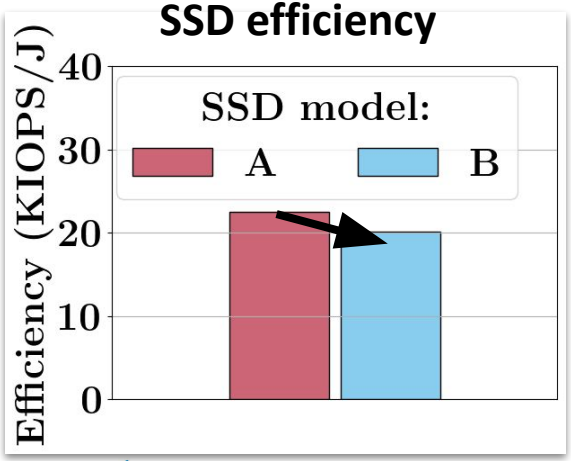


Efficiency characterization [3/3] - Attribution

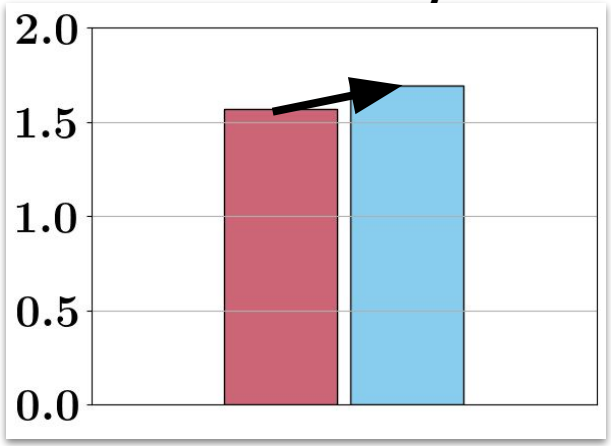
We change SSD from “A” to “B” for the varmail workload:

1. **If we look at the SSD** → A is more efficient.
2. **If we look at the CPU/Server** → B is more efficient...

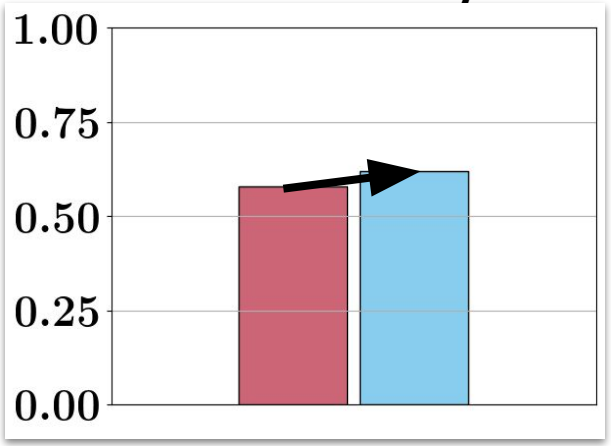
SSD efficiency



CPU efficiency



Server efficiency

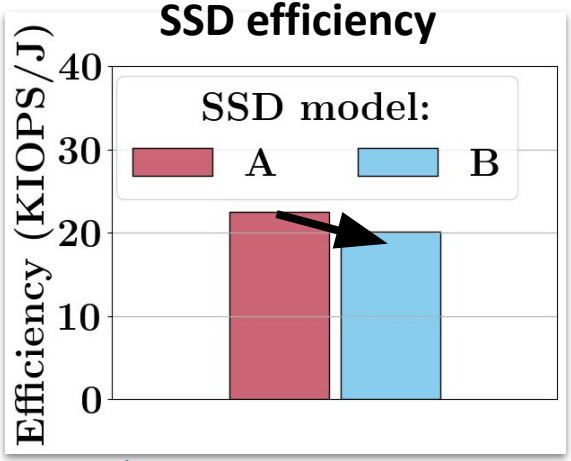


Efficiency characterization [3/3] - Attribution

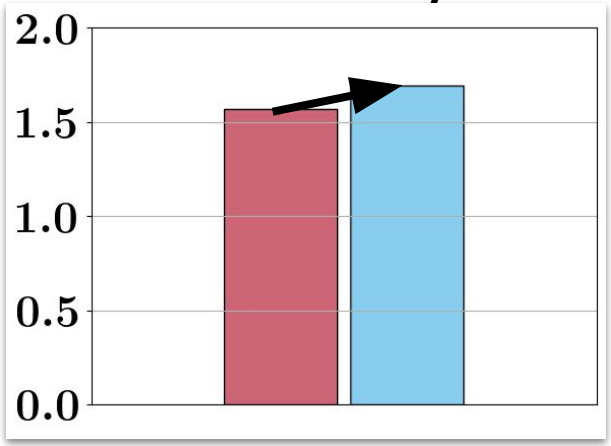
We change SSD from “A” to “B” for the varmail workload:

1. **If we look at the SSD** → A is more efficient.
2. **If we look at the CPU/Server** → B is more efficient…
3. **For efficiency analysis, we *should* measure all components.**

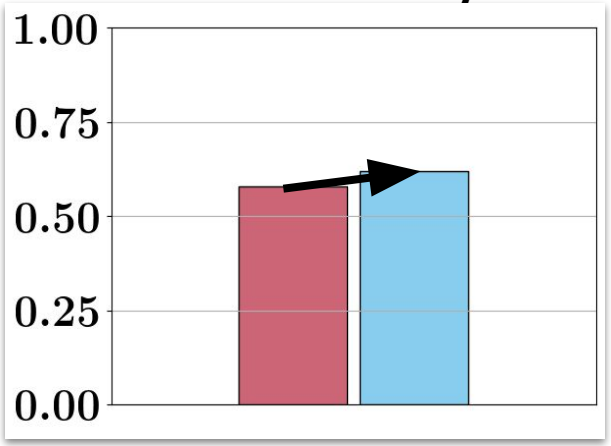
SSD efficiency



CPU efficiency



Server efficiency



More details/results in the paper…

Characterizing Energy Efficiency Trade-offs in the Linux Storage Stack for Flash-based NVMe SSDs

1st Joseph Kanichai* — *Vrije Universiteit Amsterdam*, Amsterdam, the Netherlands
2nd Krijn Doekemeijer* — *Vrije Universiteit Amsterdam*, Amsterdam, the Netherlands
3rd Steven van der Vlugt — *ASTRON (Netherlands Institute for Radio Astronomy)*, Dwingeloo, the Netherlands
4th Dante Niewenhuis — *Vrije Universiteit Amsterdam*, Amsterdam, the Netherlands
5th Tiziano De Matteis — *Vrije Universiteit Amsterdam*, Amsterdam, the Netherlands
6th Animesh Trivedi — *IBM Research*, Zurich, Switzerland

***Abstract*—Data centers (DCs) consume a large and growing share of global energy. Within DCs, storage already accounts for 5–9% of total DC energy, making storage optimization an important lever to reduce DC energy use. In this work, we focus on improving the energy efficiency of high-performance storage, where flash-based NVMe SSDs are the state-of-the-practice storage devices. To achieve this improvement, practitioners need to understand the impact of their software configurations on the efficiency of such storage. However, such understanding is limited for NVMe. Specifically, the storage stack exposes many individual knobs, while commonly conducted single source measurements further obscure which hardware component changes efficiency.**

In this work, we address this gap, enabling practitioners to understand how to make their storage workloads more energy efficient. First, we characterize the energy efficiency trade-offs for storage knobs across the Linux storage stack using 5 commercially available consumer and datacenter SSDs, including I/O access patterns, hardware-specific knobs such as DVFS and NVMe power states, I/O schedulers, I/O control, I/O interfaces, filesystems, and applications. We further attribute energy-efficiency changes separately to the SSD, CPU, and whole server. From this characterization, we make 7 observations for NVMe workloads.

Across our 7 observations, we find that maximizing I/O bandwidth generally improves SSD, CPU, and server efficiency, while configuration efficiency remain component and device specific. Second, to enable practitioners to reproduce our characterization and extend it with additional experiments on their own hardware setup, we introduce `nvme-energy-bench`, a benchmark orchestration framework. Its design is modular, allowing practitioners to add or remove sensors as needed and to incorporate custom workloads easily. All our code is open source and available at https://github.com/atlarge-research/nvme-energy-bench.

***Index Terms*—Energy efficiency, Measurements, NVMe**

I. Introduction

Data centers (DCs) are power-hungry; today, individual DCs consume as much power as cities [1], and by 2026, all DCs combined are expected to consume 2–6% of global energy [2]. This consumption needs to be reduced due to monetary costs, grid capacity, and environmental concerns [3], [4]. Reducing energy use of storage systems is especially crucial, since storage is estimated to account for 5–9% DC energy [5], [6]. Moreover, in storage-dense deployments such as in flash arrays (e.g., containing 16–24 NVMe SSDs), the aggregate SSD power can reach 12–15% of system power while the CPU consumes as much as 36% [7], making storage energy optimization particularly impactful. This work is part of this concerted storage effort and focuses on the *energy efficiency of high-performance data storage*, where flash-based NVMe SSDs are the state-of-the-practice. To make such state-of-the-practice storage more efficient, practitioners need to understand which software knobs in the storage stack affect efficiency trade-offs.

Nevertheless, the trade-offs for NVMe SSDs in Linux are largely understudied, and different SSD models can exhibit different characteristics. A key reason for SSD models differing is that they are black-box devices: their firmware-controlled internals (e.g., flash translation layer (FTL) mappings, garbage

*These authors contributed equally to this work This work is partially supported by Netherlands-funded projects NWO OffSense (OCENW.KLEIN.209), NWO MLS (OCENW.KLEIN.561), and National Growth Fund project 6G FNS. The work is also supported by EU-funded projects MSCA CloudStars (g.a. 101086248) and RADIOBLOCKS (g.a. 101093934). K. Doekemeijer is funded by the VU PhD innovation program.

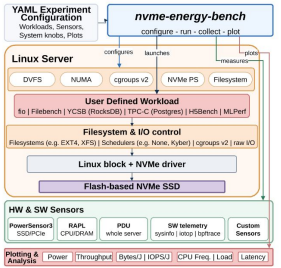


Fig. 1. Holistic overview of `nvme-energy-bench`: its workload & system configurations, its HW & SW sensors, and automated plotting & analysis.

TABLE II. SSD specifications (vendor reported numbers)

SSD	*Capacity*	*DRAM cache*	*Random R/W IOPS*	*TBW*	*Purpose*
"A"	1 TB	HMB	1.5 M / 1.5 M	600	Consumer
"B"	1 TB	1 GB	1 M / 1 M	600	Consumer
"C"	960 GB	-	413 k / 44 k	1400	Datacenter
"D"	960 GB	Avail	580 k / 70 k	1752	Datacenter
"E"	3.84 TB	Avail	1 M / 200 k	7000	Datacenter

SSD vendors: Samsung, Western Digital, Solidigm, Crucial.

leading to **7** observations. These recommendations can help practitioners understand the factors affecting NVMe energy efficiency and optimize their workloads on SSDs.

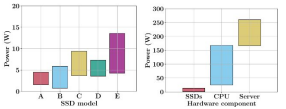
- All our code, including `nvme-energy-bench`, is open source and available at https://github.com/atlarge-research/nvme-energy-bench.

II. Setup and Approach

Below, we detail our hardware, software, and measurement setup, and how we evaluate energy efficiency. We run all benchmarks with `nvme-energy-bench`, and the discussed benchmarks are readily available as YAML configurations.

Hardware and measurement setup: For SSD power measurements, we use the state-of-the-art PowerSensor3 [23], a custom, open source, and accurate PCIe power sensor. For server-level measurements, we use two NETIO PowerPDU 4KS [24] (two inlets). For the CPU and DRAM, we use the software-based RAPL [25]; throughout this work, we will refer to this setup as CPU. We sample from the PowerSensor3 and RAPL at 1 ms granularity and from the PDUs at 500 ms granularity (minimum granularity). We detail our server hardware in Tab. I. We evaluate all our workloads on 5 anonymous, representative NVMe flash SSDs from major SSD vendors (Tab. II) to show the generalizability of our results. SSDs "A" and "B" are intended for lower power consumer devices and support *autonomous power state transition* (APST). APST lets the SSD move between power states autonomously, using a table of idle time thresholds. SSDs "C"–"E" are intended for data centers (no APST). The *DRAM cache* column in Tab. II indicates the DRAM cache used, usually for the purpose of caching FTL mapping tables and absorbing writes [26], [27]. SSD "A" has no on-device DRAM and instead uses *host memory buffer* (HMB), an NVMe feature where the SSD borrows a small region of host DRAM over PCIe. The datacenter SSDs "D" and "E" have on-device DRAM but the vendor does not publish its size, so we mark these as "Avail". These differences matter for energy because DRAM caching affects idle power, FTL lookup cost, and write buffering behavior, all of which interact with the knobs we study. We evaluate one SSD at a time to isolate and attribute energy changes to the SSD, CPU and whole server independently. Multi-SSD setups, RAID, and networked/distributed storage systems introduce various additional knobs and additional host and network overheads that confound attribution. We discuss their implications and directions for future work in §V.

To understand the power range of our hardware setup and thus the power that can be saved, we plot the power ranges



(a) Range of all 5 SSDs (b) Range of all hardware; note server PDUs are sampled at 500 ms

Fig. 2. Observed power range of our hardware setup.

of the used hardware components in Fig. 2. The SSD ranges are inferred from all our micro-benchmark evaluations (see §III). The CPU (RAPL) and server range (PDU) are inferred by measuring idle power with the lowest evaluated DVFS frequency (see §III), our micro-benchmark evaluations, and by running the *stress-ng* [28] "cpu" benchmark on all cores with the highest evaluated DVFS. For this plot, we use a 100 ms rolling average for both CPU and SSDs to remove noise for the minima and maxima. The server measurements are at 500 ms, since we measure an AC signal and the PowerPDU is limited to this granularity. We hypothesize that the PDU's observed range is narrower than the CPU's because the PDU measures AC power at the wall (before the PSU), capturing the PSU's conversion overhead and any smoothing inherent to the AC measurement at its coarse minimum sampling granularity of 500 ms. As a result, transient power variations of individual components are attenuated in the PDU signal. RAPL, being a hybrid model (combining voltage regulator data with a software model), while very accurate, may also report a slightly wider range than the true electrical draw [25]. We additionally verified the PDU readings using IPMI DCMI.

Software: We evaluate on Ubuntu 22.04.5 LTS with Linux kernel 6.17.3. In our experiments, we use `nvme-energy-bench` integrated with various benchmark tools. For micro-benchmarking, we use the state-of-the-practice *fio* [29] (3.41, commit #ed675d3).

For application benchmarks, we use *filebench* [30] (1.4.9.1, #0ce9491), and *YCSB* [31] (0.18.0, #1e62880) with RocksDB (6.2.2). As summarized in Fig. 1, each experiment is configured with YAML files for benchmarks, sensors, system knobs, and plots. For each run, `nvme-energy-bench` prepares the NVMe SSD and file system; sets the NVMe power state; configures DVFS, NUMA, schedulers, and cgroups; and collects hardware and software sensor data with a workload such as *fio*, *filebench*, or *YCSB*. Additionally, it repeats workloads for standard deviation, and automates analysis and plotting.

Experimental design: A combinatorial sweep of all knobs is prohibitively large, and we do not attempt an exhaustive search. We use prior work and preliminary characterizations with a single knob at a time to establish relevant ranges and saturation points. We then re-evaluate representative points in cross-knob experiments. We collect cross-knob data for request sizes, queue depths, I/O engines, DVFS, NVMe power states,

Future work

1. **Autotuned I/O engine:** An I/O engine that can use known energy optima information to tune I/O for increased efficiency based on workload
2. **More hardware variation:** CPU and server configurations
3. **Many SSDs:** Extend to Multi-SSD/RAID setups
4. **More knobs:** Look into CPU C-states
5. **Modify applications/cgroups**: use DVFS/NVMe PS (both promising techniques, but typically not used by default)

Take-away message

1. **Problem:** DC storage is power-hungry
   - NVMe flash SSDs are SOTP for fast storage
   - Energy efficiency trade-offs are challenging: black box, component-level attribution
2. **Solution:**
   - Use nvme-energy-bench, our benchmark orchestrator
   - Characterizing trade-offs per component and knob is key to understanding efficiency
3. **Efficiency findings:**
   - Bandwidth is key to efficiency
   - I/O shape, access pattern, I/O interface, DVFS also impact efficiency

Paper: https://atlarge-research.com/pdfs/2026-iiswc-nvme-energy.pdf
Code: https://github.com/atlarge-research/nvme-energy-bench

Supported by NWO, 6G FNS, MSCA CloudStars, and RADIOBLOCKS

Further reading from our team

1. Steven van der Vlugt, Leon Oostrum, Gijs Schoonderbeek, Ben van Werkhoven, Bram Veenboer, Krijn Doekemeijer, John Romein, *PowerSensor3: A Fast and Accurate Open Source Power Measurement Tool*. 2025 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). 2025
2. 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
3. Zebin Ren 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'23). 2023.
4. Krijn Doekemeijer, Zebin Ren, Tiziano De Matteis, Balakrishnan Chandrasekaran, and Animesh Trivedi. *Does Linux Provide Performance Isolation for NVMe SSDs? Configuring cgroups for I/O Control in the NVMe Era?.* In 2025 IEEE International Symposium on Workload Characterization (IISWC). 2025.
5. 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.
6. 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