PASS: A Power Adaptive Storage Server

Dedong Xie
University of Washington
Seattle, USA
dedongx@cs.washington.edu

Theano Stavrinos*
Databricks
Seattle, USA
theano.stavrinos@gmail.com

Jonggyu Park
University of Washington
Seattle, USA
jonggyu@cs.washington.edu

Simon Peter
University of Washington
Seattle, USA
simpeter@cs.washington.edu

Baris Kasikci
University of Washington
Seattle, USA
baris@cs.washington.edu

Thomas E. Anderson
University of Washington
Seattle, USA
tom@cs.washington.edu

Abstract

Power management has become important in data centers. Since data center workloads are often dynamic, it is common practice to conserve energy by scaling resources up or down to match the workload. And since data centers often oversubscribe the power delivery infrastructure, operators can add power capping on top of these workload-proportional systems to adjust to available power. We find that this combination leaves performance on the table and provides only a limited power control range. Instead, we argue for *power-adaptive* systems that attempt to make the best use of the available power budget. To illustrate this approach, we built PASS, a power-adaptive storage system. PASS considers the interactions between different system components, including software and hardware, when making its power management decisions. For example, when under the same power constraint, PASS achieves 3–25× better throughput on filebench workloads than Intel's SPDK storage stack with Google Thunderbolt, a state of the art power capping system.

CCS Concepts: • **Hardware → Enterprise level and data centers power issues**; • **Information systems → Storage power management**.

Keywords: Data center, storage system, power management

ACM Reference Format:
Dedong Xie, Theano Stavrinos, Jonggyu Park, Simon Peter, Baris Kasikci, and Thomas E. Anderson. 2026. PASS: A Power Adaptive Storage Server. In *21st European Conference on Computer Systems (EUROSYS '26), April 27–30, 2026, Edinburgh, Scotland UK*. ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3767295.3769354

*Work done at the University of Washington.

EUROSYS '26, April 27–30, 2026, Edinburgh, Scotland UK

ACM ISBN 979-8-4007-2212-7/26/04
https://doi.org/10.1145/3767295.3769354

1 Introduction

Power management has become a critical challenge for data centers. To reduce both capital costs and operational energy loss, operators often oversubscribe power to accommodate higher loads within existing infrastructure [27]. As demand surges in some parts of the data center, other parts must quickly scale back so that the overall system remains within the power system's physical limits. Additionally, in some regions, data center operators may be required to participate in grid demand-response programs, requiring them to reduce power usage at times of high demand. Disruptions, such as natural disasters or internal power infrastructure failures, can further constrain available power. A large power control dynamic range, with minimal application performance overhead, is thus desirable.

Today, data center operators typically combine *workload-proportional* systems with *power capping* on top to stay within power budgets [27]. *Workload-proportional* hardware and software allocates resources and, accordingly, draws power in proportion to workload intensity (*load*). For instance, the interrupt-based Linux kernel storage stack is workload-proportional. It puts cores to sleep when there is no work, while interrupts wake up cores as requests arrive. *Power capping* systems, like Google's Thunderbolt [27] (via software CPU bandwidth control), and hardware CPU and device power capping [12, 18, 22], operate on top of these workload-proportional stacks, limiting resource use to keep server instantaneous power draw within a specified budget.

Unfortunately, this design limits both achievable performance and power control dynamic range. As we will show in §2.3, workload-proportional stacks introduce inefficiencies whenever their mechanisms reallocate resources. As this happens at the frequency of workload fluctuations, it degrades performance for typical bursty data center workloads. At the same time, power capping systems are not integrated with the stacks they control and introduce further overheads when goals are at odds. For example, a workload-proportional stack may allocate additional threads to handle a higher load, while the power capping system limits available CPU cycles to stay within a power budget. This yields a suboptimal configuration, where many threads may be

context-switched over just a few cores. Finally, these systems often do not coordinate power control mechanisms, leading to a diminished power control dynamic range. For example, focusing only on CPU power control may leave a large fraction of other power-hungry devices uncontrolled (cf. §2.2).

In this paper, we argue that data center systems should instead be *power-adaptive*. Rather than drawing power proportional to load and relying on reactive power capping, power-adaptive systems proactively coordinate their components to operate within a dynamic but relatively (compared to workload variability) stable power budget. Power adaptivity leverages some of the same mechanisms as workload-proportionality, such as disabling cores or throttling bandwidth, but does so with the goal of maximizing application performance under power constraints. Power adaptivity combines goals of prior work on intermittent power [40], which is budget driven, and coordinated power control [31, 32], which aims to expand the power control dynamic range by adjusting resource usage across the system stack.

As a case study of power-adaptive system design, this paper presents PASS, a power-adaptive all-flash storage server (AFS). Three trends make flash-based storage servers interesting for studying power adaptivity. First, demand for storage is exploding due to data-intensive AI workloads, making storage systems prime candidates for power adaptivity [5, 38, 59]. Second, flash-based SSDs, which are increasingly replacing HDDs [55], offer a growing and at least 2× wider power dynamic range than HDDs [59] (cf. §2.2). Third, storage disaggregation, which is increasingly common [4, 57], allows us to manage the power drawn for storage separately from other parts of the data center.

To maximize performance under a power budget, PASS employs a closed-loop online control system that takes the server's current power draw and a power budget as inputs. It then determines work assignments to CPUs and SSDs in a coordinated fashion. Specifically, PASS controls four components across the storage stack: CPU allocation, CPU power limit, CPU bandwidth, and SSD bandwidth. To do so, PASS relies on CPU and SSD power models derived offline in a profiling step for the current machine configuration. Because performance is not a linear function of power, particularly under tight power constraints, PASS must sometimes turn away work or reduce service to remain within budget, highlighting the need for intelligent control strategies that prioritize bottleneck resources.

We make the following contributions:

- We present a power and performance study of AFS designs. We find that power capped workload proportional designs deliver limited application performance and a narrow power dynamic range.
- Based on our performance study, we design, implement, and evaluate PASS, a power-adaptive storage server. PASS maximizes utilization of a given power budget to maximize storage system performance, while controlling CPU and SSD work assignments in concert across the storage stack to maximize the power control dynamic range.
- On a storage server with 10 SSDs and 8 CPU cores, we conduct a thorough study of PASS's performance and power consumption using microbenchmarks and application benchmarks of file systems, databases, and key-value stores. For example, PASS achieves 3–25× better throughput, with greater relative performance when power is more constrained, across several filebench workloads relative to a power-proportional setup of SPDK [43] capped via Google Thunderbolt [27], a state of the art power capping system.

2 Motivation

Power is an increasingly constrained resource in modern data centers, driven by power infrastructure limitations, participation in grid demand-response programs, and the power-variable nature of emerging workloads such as AI training. In this section, we examine the implications of these constraints for storage systems. We begin by discussing the growing operational and infrastructural pressures that data centers face in managing power (§2.1). We then highlight why storage systems, as major power consumers and increasingly disaggregated components, must become more power-adaptive to meet data center-wide power goals without sacrificing performance (§2.2). Finally, we present an empirical study of existing all-flash storage server (AFS) designs, which predominantly follow workload-proportional principles and rely on power capping mechanisms to enforce budgets. Our findings show that these approaches introduce overhead, fail to coordinate control across components, and ultimately leave performance on the table under tight power constraints (§2.3). These insights motivate the need for explicitly power-adaptive storage systems that optimize resource usage based on power availability, rather than merely reacting to workload-driven demand.

2.1 Data center Power Constraints

Modern data centers increasingly face strict power constraints [35]. The rise of AI workloads has led to sharp, rapid power swings due to the bulk-synchronous parallel nature of model training. These fluctuations must be buffered to avoid destabilizing the grid. In some regions, data center power demand is expected to account for 30% of local grid capacity [29]. As a result, data centers often participate in grid demand-response agreements that require temporary reductions in power usage. To maximize infrastructure utilization and minimize costs, operators also employ power over-subscription—typically 20–30% above rated capacity—which necessitates rapid reductions in power draw during usage spikes to avoid tripping circuit breakers [27]. Other

sources of power variability include diurnal patterns in renewable generation and extreme weather events [35].

To remain operational, data centers must respond to power availability changes in tens of seconds—a timescale dictated by the thermal and electrical limits of power infrastructure such as breakers and transformers [58]. These realities create a need for data center components that can finely and quickly modulate their power draw.

2.2 The Need for Storage Power Adaptivity

Storage systems account for 18–21% of total data center power consumption [3, 11, 38] and are increasingly built as disaggregated services [1, 24, 33]. Modern storage servers frequently consist of high-performance all-flash configurations built from NVMe SSDs. With a typical active power draw of 25W versus 5W at idle [39, 42] and an announced active power draw of 75W [16, 41], flash-based SSDs offer a growing and at least 2× wider per-unit power dynamic range than HDDs that operate at 12W when active and 3W when idle [13, 50, 54]. SSDs also use more power than any other component in an all-flash storage server. For example, in our AFS setup with 10 SSDs, the SSDs account for 250W at peak, while the CPUs consume 200W, and the remaining components account for 120W. Together, the CPUs and SSDs comprise almost 80% of total AFS power (cf. Figure 7) and they are the only components with meaningful power dynamic range and controllability. Other components (e.g., network interface cards, fans, and motherboard) either consume a constant amount of power or their consumption is indirectly tied to compute and storage IO activity.

As data centers adopt fine-grained, system-wide power management, the ability of the storage subsystem to dynamically scale its power thus becomes critical. Without power-adaptive storage, other systems must absorb the burden of meeting power caps, often resulting in disproportionate performance degradation. In contrast, a power-adaptive storage system that can, for example, cut its power consumption by 50% can entirely absorb a 10% data center-wide power reduction, allowing other systems to continue operating at full performance. Because storage availability directly affects the operability of dependent workloads, any power adaptation must preserve responsiveness. The wider a power control dynamic range a storage system can support without degrading responsiveness, the more effectively it can contribute to data center-wide power goals.

These constraints and opportunities motivate the need for explicitly power-adaptive storage systems—systems that adjust their power draw in response to a specified power budget rather than passively scaling with workload demand.

2.3 Existing Storage Stacks Fall Short

Today's AFS storage stacks are *workload-proportional*, meaning they draw power in proportion to the amount of load the system handles. This design is widely adopted because it naturally aligns energy use with system activity, improving energy efficiency during low-utilization periods. For example, the interrupt-based Linux NVMe-oF target is workload-proportional because the CPU generally only performs work when there are requests to process, i.e., when an IO request is submitted to the disk, or a storage device completes work and sends an interrupt to the kernel. Polling-based frameworks can also be workload-proportional by consolidating polling threads onto fewer CPUs when load is low. For example, the Storage Performance Development Kit (SPDK) NVMe-oF target has a workload-proportional dynamic scheduler, which reduces the number of polling cores when load is low [43].

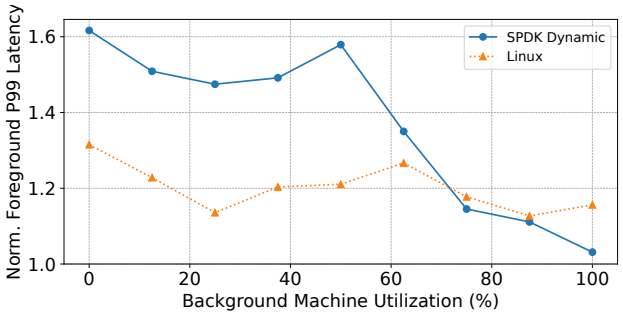
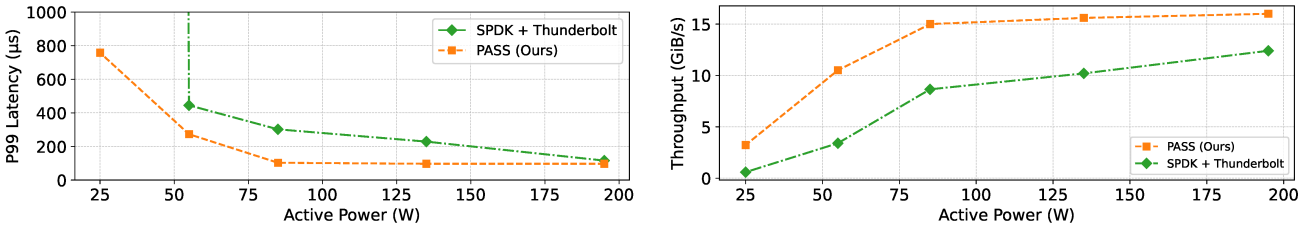


Figure 1. Foreground random 4 KiB write tail latency, normalized to static baseline, varying background job intensity.

Inefficiencies of workload-proportionality. While effective for energy savings, workload-proportional systems introduce inefficiencies when reallocating resources under fluctuating load, especially in low-utilization scenarios common in data centers [49].

These systems depend on workload telemetry to guide resource allocation, requiring a control interval. If the interval is too long, rapid workload variation is underestimated; if too short, frequent reconfiguration incurs high overhead. Bursty workloads make this tradeoff particularly problematic, since no single interval fits all burst patterns. Further, interrupt-based stacks such as Linux incur additional overhead from frequent context switches as interrupts wake sleeping cores. Combined, control delays and overheads make workload-proportional designs inefficient for bursty, variable datacenter loads.

To demonstrate this, we experimentally evaluate (cf. §5 for the experimental setup) the responsiveness of a mixed storage workload consisting of a constant background load and a periodic foreground burst across both the Linux and SPDK workload-proportional targets and compare to a static baseline that keeps all system resources allocated. The background workload comprises eight `fio` [2] jobs issuing 4 KiB random writes to 9 of the 10 SSDs. The foreground burst consists of 16 synchronized `fio` jobs performing 4 KiB random writes to the remaining disk for 100 ms every 5 seconds. The burst duration is similar to the resource reallocation



(a) Random write 4 KiB foreground tail latency.

(b) Random write 32 KiB background throughput.

Figure 2. Performance of SPDK + Thunderbolt and PASS under different power budgets.

time of the workload-proportional system. We collect the 99th percentile latency of the foreground workload at the initiator, as well as target server utilization. Each experiment runs for 3 minutes, and metrics are collected after performance stabilizes. Figure 1 shows that workload-proportional stacks exhibit up to 1.6× higher tail latency for bursty work compared to a static baseline, at server utilization below 50%. As background workload intensity increases, workload-proportional systems begin to catch up in performance, as more resources are provisioned overall, indirectly benefiting the foreground workload.

Power capping. On top of these storage stacks, power capping systems are employed to constrain power consumption within a set budget. Current power capping mechanisms typically cap the power of individual system components, such as CPUs or SSDs, independently.

CPU power can be limited using both hardware and software mechanisms. Intel's Running Average Power Limit (RAPL) enforces socket-level power caps in hardware [18]. It combines dynamic voltage and frequency scaling (DVFS) with fine-grained feedback control on core scheduling. Unlike conventional DVFS, which offers limited control levels, RAPL provides more precise and responsive power regulation. At the software level, CPU bandwidth control limits the fraction of time a CPU can be active, thereby reducing its power draw [21]. This mechanism is used in systems such as Google's Thunderbolt [27]. Another approach is CPU jailing, which restricts the number of cores available to an application, effectively reducing total power consumption.

Many modern SSDs include a power-capping feature in hardware, as described in the NVMe specification [12]. Such disks have multiple power states, preset by the manufacturer. Each power state caps an SSD's power draw below a threshold over a time window, typically 10 seconds [12]. These power states are operational, i.e., the SSD remains responsive to incoming IO requests. Software disk bandwidth control is another mechanism for controlling SSD power draw, since SSD power draw is proportional to load [54].

Power capping further degrades performance. While existing power capping mechanisms are effective at enforcing power budgets at the component level, they operate in isolation and lack coordination with system-wide workload behavior. These mechanisms often conflict with workload-proportional strategies, leading to inefficiencies and degraded performance under constrained conditions. Critically, they do not reason about which component is the current performance bottleneck, nor do they adapt allocations based on end-to-end system impact. This disconnect limits both the performance and the usable dynamic power range of AFS systems.

We demonstrate this behavior experimentally by comparing a mixed workload's background (5 jobs of 32 KiB random writes) throughput and foreground (16 jobs of 4 KiB random writes 100 ms bursts every 5 seconds) latency with either SPDK workload-proportional storage stacks power-capped with Thunderbolt or PASS, which is power adaptive. Linux does not support a power capping system and we cannot compare to it in this configuration.

Figure 2 shows that PASS consistently outperforms SPDK+Thunderbolt, in particular under lower power budgets. In terms of tail latency (Figure 2a), PASS outperforms SPDK+Thunderbolt between 1.2×–2.9× under active power budgets above 60 W. Thunderbolt fails to provide adequate tail latency for active power budgets below 60 W. In terms of throughput (Figure 2b), PASS outperforms SPDK+Thunderbolt between 1.3× under high power budgets, up to 5.5× under low power budgets.

Conclusion. Our evaluation shows that existing AFS systems built around workload-proportional designs deliver sub-optimal performance under constrained power budgets. When paired with isolated, single-component power control mechanisms, these systems fail to maintain high performance across a wide dynamic power range. In contrast, manually configured, load-agnostic setups are able to meet stringent power constraints while achieving better performance. These findings underscore the need for cross-layer, power-adaptive designs that explicitly coordinate power control across components to maximize performance under power constraints.

3 PASS Design

PASS is a power-adaptive all-flash storage (AFS) server with two design goals:

1. **Maximize performance under a dynamic power budget.** The system should dynamically allocate resources to achieve the best possible performance while adhering to the given power constraints. The adjustment needs to be fast and accurate to prevent physical damage to the power infrastructure.
2. **Maximize power dynamic range.** By extending the range of power adaptability, PASS can remain operational across a broader spectrum of power budgets, making it suitable for utility demand-response.

Based on our observations from Section 2.3, we establish two key principles to guide the design of PASS:

1. **Use the entire power budget.** Fully utilizing the available power budget enables PASS to maximize resource allocation and improve overall performance.
2. **Manage power across multiple storage system components.** Effective power management requires a cross-layer approach that incorporates mechanisms at multiple levels, including the CPU, disk bandwidth, and other hardware and software layers. As shown in our experiments, relying on a single mechanism, such as CPU hardware power capping, limits both the power dynamic range and performance potential under power constraints.

These principles provide a framework for achieving PASS's design goals and we employ them in control algorithms and IO mechanisms. While PASS is designed to attempt to utilize the full available power budget, even when performance gains may be marginal (cf. §5.3 for a discussion of energy efficiency across scenarios), this choice reflects our goal of studying a power-adaptive design in its pure form. In practice, a data center-wide power control plane would set appropriate storage power budgets, based on broader energy and performance trade-offs. Integrating PASS with such a control plane is an important direction for future work.

In this section, we first provide a high-level overview of PASS's architecture (§3.1), detailing how it integrates with the storage system to manage power dynamically. PASS leverages power-performance models of CPUs and SSDs, the two major power consumers in AFS, created via offline system profiling (§3.2). An online controller then dynamically adapts the AFS configuration to maximize performance while meeting constraints, using the power model (§3.3).

3.1 PASS AFS Design Overview

PASS operates within a disaggregated storage architecture. PASS hosts NVMe SSDs as storage devices and is built on Intel's SPDK as the underlying storage driver. Hence, like SPDK, PASS is polling-based. PASS uses the NVMe-over-Fabrics (NVMe-oF) protocol with RDMA, enabling PASS to act as a network storage target (server) for initiators (clients) that host virtual machines (VMs). Through NVMe-oF, the physical SSDs in the storage target are exported as virtual NVMe devices. These devices are then presented to initiators, for use by the VMs via the initiator's hypervisor.

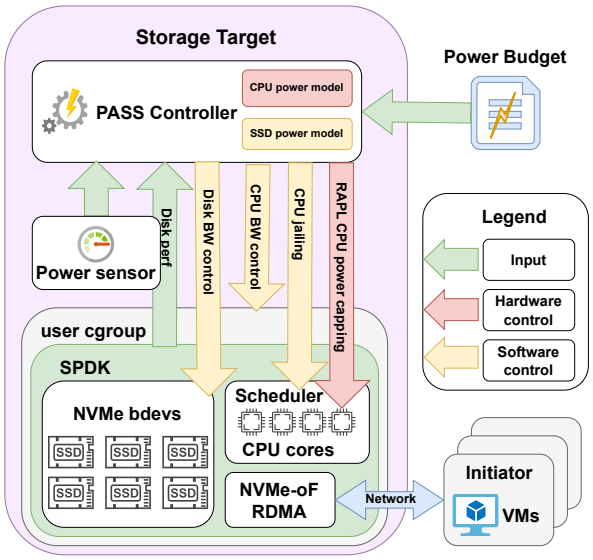


Figure 3. Overview of the PASS controller.

As depicted in Figure 3, PASS receives both power sensor and budget inputs. Power budgets are set by administrators, e.g., as a result of grid constraints, power equipment failures, or the need to allocate power to other systems. PASS periodically queries the server's power sensors to monitor power draw. The system compares the reported power consumption against the budget at a configurable frequency. We set PASS's frequency to be once every second, close to the sensor query latency, so PASS can immediately react to the server's current power draw.

PASS manages power across all AFS layers to match the server's power draw to the specified budget. When a mismatch occurs, PASS calculates adjustments to the server configuration based on its power-performance model, such as modifying CPU core assignments and disk bandwidth limits, or applying CPU hardware and software power capping mechanisms (e.g., RAPL [18] and CPU bandwidth control via user cgroups [21]). If the power usage is below the budget, PASS increases performance by adjusting the configuration to maximize the system's use of the power budget.

3.2 Power-Performance Models

Within a storage server, CPU and SSD are the power proportional components while other components are either proportional to total power (like cooling fans) or relatively stable in power consumption (like DRAM and motherboard).

Thus, to control the total power consumption of a storage server, we focus on controlling the power of CPUs and SSDs.

To enable PASS to operate effectively under dynamic power budgets, we first build a model to quantify how different configurations of CPU and SSDs impact power consumption and performance. These models predict the consequences of controller actions on power and performance in response to changing power budgets. While SSD power and throughput have a simple linear relationship that is covered nicely by existing models (§3.2.1), CPUs require offline profiling to build accurate power-performance models (§3.2.2). By characterizing the performance and power draw of various CPU configurations (e.g., number of polling cores, RAPL limits, and bandwidth throttling) and SSD workload behaviors (e.g., throughput versus power), PASS can optimize system operation without trying each possible configurations at runtime.

3.2.1 SSD Model. As found by previous work [54], SSD power and throughput have a linear relationship: across different types of SSDs of different vendors, the coefficient of determination of linear regression is higher than 0.99. We use this linear relationship as the power model for SSDs. For each available SSD power capping level, bandwidth is measured and integrated into the model. For SSDs without a power capping facility, the vendor's specification of bandwidth and power can be used and a linear model created from them.

3.2.2 CPU Model. We design a CPU profiler to find the relationship between CPU configuration, power, and performance. The job of our CPU profiler is to build a model that predicts the power and performance delivered by a given CPU configuration and, using the model, to generate a CPU configuration control policy that maximizes performance under a given power budget. The total search space of all possible CPU configurations is large as it has three dimensions: CPU number of cores, RAPL powercap level, and CPU bandwidth limit.

To accelerate model training, instead of running full end-to-end performance evaluations of PASS on storage workloads, we use the CPU polling interval as a performance proxy. The polling interval is monotonically related to end-to-end performance and can be profiled offline. As shown in Section 5.6, this approach is effective. Thus, our task reduces to finding the CPU configuration that minimizes the polling interval under a given power budget. To build the CPU model, we employ the following steps:

1. We perform a coarse-grained grid search across the three configuration axes to identify the CPU configuration that both satisfies the power budget and minimizes the 99th-percentile (p99) polling interval. Specifically, we vary RAPL power budgets in 10 W increments and exponentially increase CPU bandwidth within each budget to find the best performing configuration.
2. For each CPU configuration, the profiler records both the average elapsed CPU ticks between I/O polling events (the *polling interval duration*) on each CPU and the corresponding power consumption. Because Intel's RAPL interface provides stable CPU power measurements within 2 seconds [25], we profile each configuration for 5 seconds to obtain reliable power results.
3. When multiple configurations under a power budget achieve performance results within 5% (a *tie*), we refine the grid search with finer granularity to break the tie.
4. We repeat this process until all ties are resolved, and then fit a linear model to the resulting data.

With these techniques used to accelerate the search, we find that the total profiling time for a CPU with 32 cores, a 280W TDP with 10W power budget steps, is 8 hours. This is a one-time cost to calibrate PASS for a machine.

3.3 Online Controller

Using the offline-generated power-performance models for both the CPU and SSD, we formulate power control as an optimization problem constrained by the system's total power budget, P_{budget}. The objective is to optimally distribute P_{budget} among CPU and SSD such that we avoid reducing power to the performance-limiting component. In other words, we reduce power from the non-bottleneck resource whenever power adjustments are necessary.

Because the actual workload characteristics can shift and models may not be perfectly accurate, we implement an online controller instead of solving this optimization problem statically. The online controller operates in a *proportional control* framework to control the power draw of CPUs and SSDs, where the magnitude of a control action is proportional to the difference between the current and target system state. The controller begins with the system running at full power (i.e., no constrained power budget), and then progressively reduces CPU and SSD power to fit within the given budget.

Bottleneck-aware power allocation. As discussed in §3.2, there is a monotonic relationship between CPU performance and CPU power P_{CPU}. Likewise, SSD performance is bounded by its bandwidth, which increases linearly with SSD power P_{SSD}. Overall performance is constrained by the slower of the two components:

$$p_{\text{PASS}} = \min(A(P_{\text{CPU}}), B(P_{\text{SSD}}))$$

where A and B are monotonic transformations derived from the CPU and SSD power-performance models. To maximize system performance under a power budget, power should be reduced only from the component that is not currently the bottleneck. Our approach leverages the fact that SSD bandwidth can be directly observed and has a predictable, linear relationship with power.

The control strategy operates as follows. First, PASS reduces CPU power and monitors any resulting changes in

SSD throughput. Since SSD performance does not respond instantaneously to power changes, PASS samples throughput over five polling intervals. If the observed drop in SSD throughput exceeds the expected drop based on SSD power scaling, PASS infers that the system is CPU-bound. In this case, it reallocates power by throttling SSDs instead, using a proportional factor k derived from marginal power savings. To prevent excessive performance degradation, the new SSD bandwidth cap is set between the bandwidths observed before and after CPU throttling. Once SSD throttling is applied, the CPU configuration is restored and a new control cycle begins. PASS continuously monitors power consumption and performance metrics to determine whether additional adjustments are necessary.

This decision process is formalized in Algorithm 1. The function `get_CPU_configuration(target_power)` queries the offline CPU power-performance model to return the CPU configuration that minimizes the 99th percentile polling interval while satisfying the target power constraint. By dynamically shifting power between the CPU and SSD based on real-time performance impact, PASS preserves headroom for the bottleneck resource, maximizing power budget utilization and improving overall system throughput. We study the performance of this approach in §5.7.

Algorithm 1: Bottleneck-aware power allocation.

```
Function adjust_CPU_power(power, target_power):
    power_margin ← power − target_power
    cpu_configuration ← get_CPU_configuration (target)
    old_bw ← get_SSD_bandwidths (past 5 intervals)
    /* Try to change CPU power budget */
    set_CPU_configuration (cpu_configuration)
    new_bw ← get_SSD_bandwidths (5 intervals)
    diff_bw ← old_bw - new_bw
    if power_margin < get_SSD_diff_power (diff_bw) then
        /* Use SSD bandwidth control */
        set_SSD_bandwidths (new_bw + k × diff_bw)
        reset_CPU_configuration ()
```

Stability. Storage system power draw may be delayed after applying configuration changes due to delays in IO completion and slow (on the order of seconds) power sensor readings for system-wide power. To address this, PASS introduces *control stability thresholds* to avoid unnecessary adjustments while ensuring stable performance. PASS applies different power control policies depending on the current power draw p in relation to the power budget and two thresholds:

1. $p >$ **high threshold:** In this case, PASS aggressively reduces power usage by decreasing CPU core allocations, applying RAPL power caps, or throttling CPU and disk bandwidths. Data center power supplies can tolerate overload for a limited time. We set the high threshold 2% above the power budget, according to [27]. This configuration provides sufficient stability to keep the system within the budget, while ensuring convergence to a steady state within 10 seconds, the typical power endurance period in many data centers [27, 58].
2. **High threshold** $\geq p >$ **power budget:** PASS adjusts power usage by more gradually applying power controls.
3. **Power budget** $\geq p >$ **no-action threshold:** PASS takes no power control action to avoid unnecessary configuration changes when power draw is close to the budget. We set this threshold to 1% below the power budget.
4. **No-action threshold** $\geq p$**:** PASS aggressively allocates more resources to the system to improve performance by proportionately increasing the CPU power budget and lifting the SSD bandwidth limit. The system will then enter another cycle of feedback control to meet the power budget.

Priority-aware SSD throttling. Storage clients may be assigned different priorities to reflect their quality-of-service requirements. PASS leverages these priorities to apply differentiated SSD bandwidth throttling across virtual disks. When power needs to be reduced, PASS selectively limits bandwidth for lower-priority clients, freeing up power headroom for higher-priority workloads. This approach allows PASS to enforce power budgets while preserving the performance of critical applications.

4 Implementation

PASS is implemented on top of Intel's Storage Performance Development Kit (SPDK) [43], which offers user-space NVMe drivers and supports NVMe-over-Fabrics (NVMe-oF) via RDMA. SPDK adopts a polling-based I/O model with an adjustable number of cores that continuously poll NVMe devices and network interfaces for requests and completions.

Offline profiler. We implement the PASS offline CPU profiler in 500 lines of `C` to accurately capture polling behavior representative of SPDK's runtime performance. Since the analysis of the collected data is not performance-critical, we implement the processing pipeline, including extraction and control policy generation, in Python and shell scripts with about 900 lines of code.

For SSD power profiling, we use the methodology outlined in Xie *et al.* [54]. We construct a dedicated testbed that physically measures SSD power consumption under a range of synthetic workloads, enabling accurate modeling of power-to-performance relationships.

Online controller. The online controller integrates with SPDK's CPU scheduler to control the number of polling cores. It uses the `thread_set_cpumask` RPC call of SPDK to set a CPU mask on the CPUs to run polling threads. It also uses Intel RAPL for CPU power limiting, applies Linux cgroups for CPU bandwidth control, and utilizes SPDK's built-in bandwidth control of block device for SSD bandwidth throttling. It

continuously monitors system power consumption through sensors accessed via the IPMI-DCMI interface [17].

The primary constraint on control reaction time arises from the latency of power measurements. IPMI-DCMI typically requires up to 1.3 seconds to respond to a power query, which limits how frequently PASS can make control decisions. With access to faster and more responsive power sensing hardware, PASS could support higher control frequencies and finer-grained power management. Since PASS control reaction time is bounded by the sensor, we implement the online controller in 350 lines of Python.

5 Evaluation

We evaluate PASS to assess its performance, generality, and robustness. We address the following key questions:

1. By how much can PASS improve tail latency and throughput of storage IO microbenchmarks across machine configurations? How much does each PASS mechanism contribute? How is power adjusted across machine components? (§5.1)
2. By how much can PASS improve the performance of application benchmarks under various power budgets compared to baselines? (§5.2)
3. How energy efficient is PASS? (§5.3)
4. Can PASS provide performance isolation? (§5.4)
5. Can PASS generalize across different machines? (§5.5)
6. Is the IO polling interval a good performance proxy? (§5.6)
7. Is PASS's power control stable? (§5.7)

Experimental setup. All systems in our testbed use NVMe-over-Fabrics (NVMe-oF), with an *initiator* server issuing storage I/O requests to a *target* server over remote direct memory access (RDMA) [10]. The target is a Supermicro SYS-111C-NR server equipped with an Intel Xeon Gold 6430 processor, 256 GiB of DDR5 memory, and running Ubuntu 24.04 with Linux kernel version 6.8.0. It contains 10 Solidigm D7-P5510 SSDs, each with 3.84 TiB of capacity, and is outfitted with both 100 GbE and 200 GbE RDMA NICs. The initiator, a separate machine running the Linux NVMe-oF client stack, generates all I/O requests. The generation of requests is multi-process, simulating the existence of multiple concurrent applications accessing the target server. Power is measured at the target server using IPMI-DCMI [17]. We use total system power and budgets in this section. For reference, the test server's idle power consumption is 245W.

Baselines. The workload proportional baselines are 1) the interrupt-based Linux storage stack with the default NVMe-oF subsystem driver and 2) polling-based SPDK using its dynamic scheduler to control the number of cores to match the workload. SPDK experiments use its default configuration, initially allocating 8 polling cores as that achieves peak system performance. We run PASS with the performance CPU governor enabled to remove workload-proportionality; Linux and SPDK use the schedutil CPU governor.

Our power-capping baseline is to perform CPU bandwidth throttling according to Thunderbolt's RUMD (Random Unthrottling, Multiplicative Decrease) node-level load shaping algorithm [27]. In RUMD, we randomly (with a 20% chance each second) unthrottle the CPU by increasing CPU bandwidth by 5% to stabilize control of power. We run SPDK's dynamic scheduler with this mechanism (SPDK+Thunderbolt).

Workloads. We use microbenchmarks with 16 jobs each with synchronous 128 KiB sequential write or synchronous 4 KiB random reads. Our choice of workloads covers two extremes: latency-critical small reads and throughput-oriented large writes. We use `fio` [2] with Linux's libaio [19] and psync IO engine on the client to generate these workloads. We run each experiment long enough to measure the stable state performance results. For application benchmarks, we evaluate the system's performance with a file system or as a backend for a database system. To do so, we format the server disks with the ext4 [20] file system and run the filebench [45] workloads of Varmail, Fileserver, and Webserver. We also evaluate RocksDB [46] with db_bench [47] to evaluate a database workload, as well as with YCSB [7] to evaluate a key value store. For each workload, we test with different power budgets for the server. Following the design target set by Flex [58], we consider power budget violations longer than 10 seconds intolerable. We show empty results for experiments where the system violates the budget for longer than 10 seconds. In each case, we run 10 instances of the workload generator on the initiator machine to simulate multi-tenancy. We run experiments long enough to yield stable performance and we evaluate the stable-state results.

5.1 Microbenchmarks

We begin by analyzing PASS's performance improvements using microbenchmarks designed to stress specific aspects of storage performance under varying power budgets: latency-critical small random reads and throughput-intensive large sequential writes. We examine multiple hardware configurations to further evaluate the generality of PASS's cross-layer power control mechanisms.

Latency-critical workload. Figure 4 presents the tail latency for a 4 KiB random read workload. At the highest power budget, PASS achieves a 12% reduction in the p99 tail latency compared to SPDK+Thunderbolt. As the power budget tightens, the performance gap widens: PASS exhibits between 28% and 56% lower tail latency. Only PASS makes the most constrained condition practical, delivering up to 413× lower latency.

When the system is configured with only half the number of SSDs, PASS continues to outperform SPDK+Thunderbolt. In this setup, PASS achieves between 14% and 64% lower

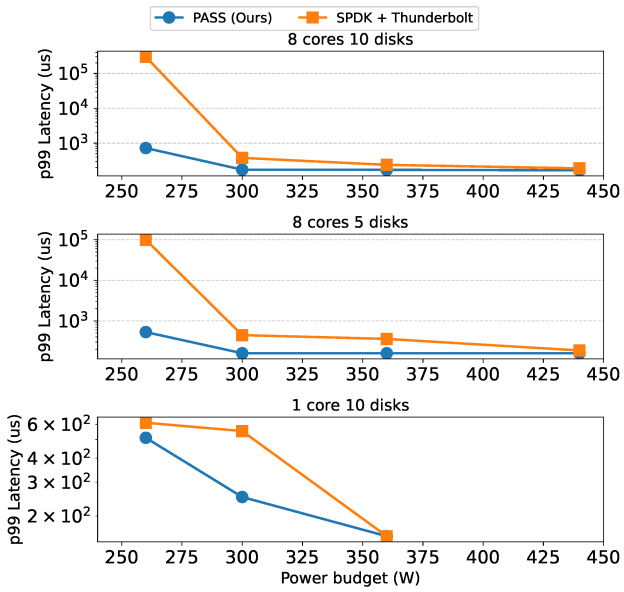


Figure 4. Average P99 latency of sync random 4 KiB reads with 16 jobs under different power budgets and server configurations.

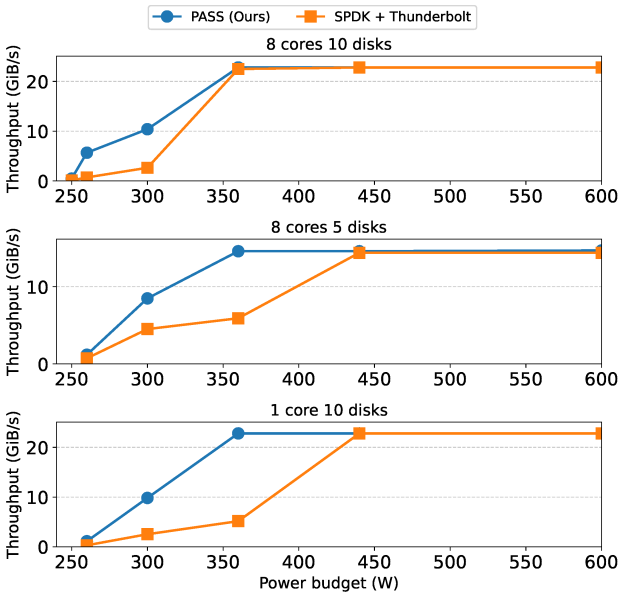


Figure 5. Throughput of sync sequential 128 KiB writes with 16 jobs under different power budgets and server configurations.

tail latency at higher power budgets, and a 189× improvement at the lowest power budget tested—just 20 W above the system's idle power of 245 W.

In the configuration with a single CPU core, where PASS cannot employ CPU jailing for power control, its power management capability is limited. Nevertheless, PASS still outperforms SPDK+Thunderbolt under the same power cap, achieving 17% to 55% lower tail latency.

Throughput-intensive workload. Figure 5 shows that PASS consistently outperforms SPDK+Thunderbolt for the 128 KiB sequential write workload, where PASS achieves 2.9× to 6.9× higher throughput.

The impact of the CPU-to-disk ratio on system performance is also evident. When fewer disks are available, the CPU accounts for a larger proportion of the system's power, requiring more aggressive power capping to meet budgets. This leads to earlier performance degradation with Thunderbolt. For example, at a 360W power budget, Thunderbolt suffers a 2.42× throughput reduction compared to PASS in the 5-disk setup. In the single-core configuration, the CPU accounts for a smaller share of total system power. As a result, SPDK+Thunderbolt must similarly throttle more aggressively to stay within the power budget. In contrast, PASS, which also manages SSD power, achieves 3.54× to 4.41× higher throughput than SPDK+Thunderbolt.

PASS performance ablation study. To demonstrate the importance of coordinated power control in achieving PASS's performance and power efficiency, we perform an ablation study. We compare PASS's full coordinated power control approach to individual control mechanisms used in isolation: RAPL power limits, CPU bandwidth control, CPU jailing, and SSD bandwidth control.

We evaluate each configuration using a synchronized 4 KiB random write workload with 16 jobs, measuring the 99th percentile (p99) tail latency. The results are shown in Figure 6. Among the individual mechanisms, CPU jailing provides the narrowest power control range and cannot reduce system power below 450 W. SSD bandwidth control achieves a lower budget of 440 W but, since it throttles only the SSDs, it results in extremely high tail latency—230.7× worse than PASS at that power level, with SSDs capped at just 10 MiB/s per disk. CPU bandwidth control offers a wider power range but underperforms RAPL in terms of latency. While RAPL matches PASS's latency at a 350 W budget, CPU bandwidth control incurs 38.9% higher p99 latency. RAPL provides the best performance and widest power range among the single-mechanism configurations, maintaining latency comparable to PASS at most budgets. However, it cannot reduce power below 290 W, and at that level, its tail latency still increases by 31.19% compared to PASS.

In summary, while individual mechanisms can partially manage power or performance, none alone achieves both the broad dynamic power range and performance of PASS. Coordinated control across CPU and SSD components is essential to fully realize the benefits of power-adaptive storage.

PASS power adjustment breakdown. We analyze how PASS adjusts power consumption across system components

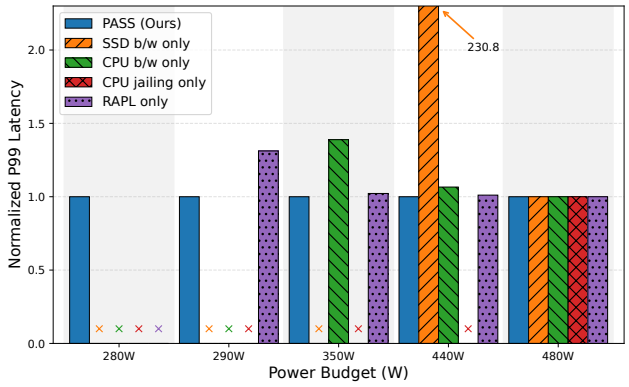


Figure 6. PASS performance ablation study. × marks configurations that exceeded the target power budget.

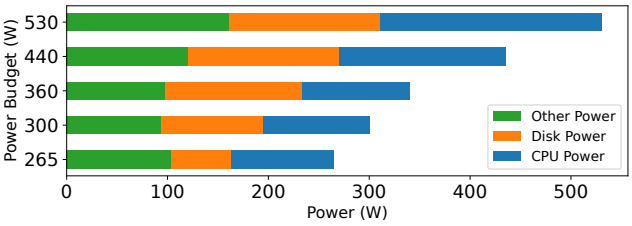


Figure 7. Power by component, varying budget.

under varying power budgets, focusing on the contributions of different hardware components—CPU, disks, and others—to the overall power savings. Using a 128 KiB sequential write workload, we evaluate power consumption at five distinct power budgets. Our analysis reveals the impact of PASS's cross-layer power management strategies, dynamically redistributing power among CPUs, disks, and other components to remain within budget. Disk power usage at each budget level is not measured but projected based on the percentage of disk bandwidth utilized. Our projection builds on prior findings that the SSDs used in our experiments are power proportional within their operational range [54] and we assume disk power scales linearly between 5W (idle) and 15W (maximum operational). Figure 7 presents the breakdown of power usage across system components. PASS sees most of its power reduction from the CPU at higher power, while at lower power the benefits come from reduction in disk power.

Figure 7 also shows that PASS achieves substantial power savings across all components: The CPU contributes about 110W to the total savings. The disks contribute about 100W, demonstrating a similar magnitude of reduction as the CPU. Other system components, such as the power needed by colling fans, contribute 55W to the overall savings.

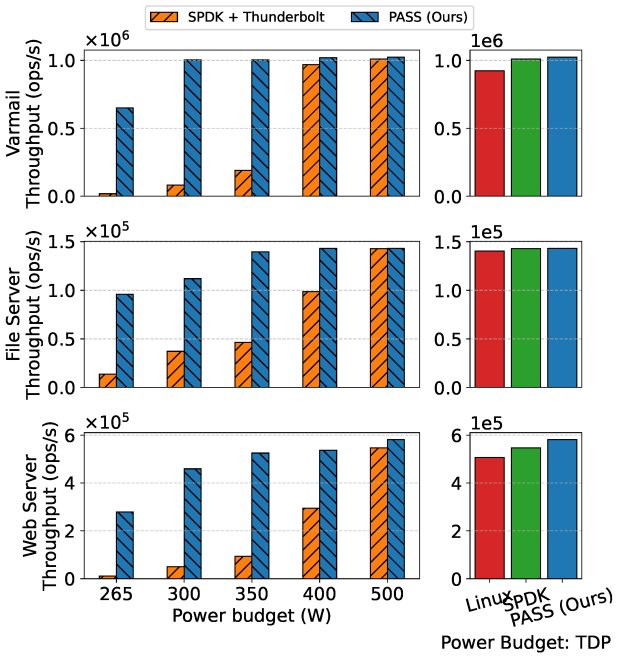


Figure 8. Throughput of Filebench benchmarks.

5.2 Application Benchmarks

We evaluate PASS across several storage-intensive application benchmarks to assess its performance under real-world workloads, comparing it against baseline systems.

File system benchmarks. To evaluate file system performance, we use Filebench with three representative workloads: Varmail, simulating an email server; Fileserver, representing file-serving operations with average file sizes increased to 1.31 MiB to reflect modern file serving trends [8]; and Webserver, modeling a lightweight web-serving workload. Each benchmark builds a file system with 200,000 files with different average directory depth, file size distribution, and file operations. Each benchmark runs for 3 minutes.

As shown in Figure 8, PASS consistently achieves the best throughput. At thermal design power (TDP), PASS outperforms Linux by 10.9%, 2%, and 14.7% respectively. PASS outperforms SPDK by 1.4%, 0.2%, and 6.3%. This highlights the superior performance of a power-adaptive stack. PASS also has superior performance under tighter power budgets. For Varmail, PASS outperforms SPDK+Thunderbolt by 3.3× at 350W and by 11.2× at 300W. For Webserver, PASS achieves 5.6× the throughput of SPDK+Thunderbolt at 350W budget and 25× at 265W budget. Even for the most demanding workload, Fileserver, PASS outperforms Thunderbolt 3× at 350W and 7× at 265W.

Database. We assess PASS in database workloads using RocksDB and four db_bench scenarios: fillseq (sequential

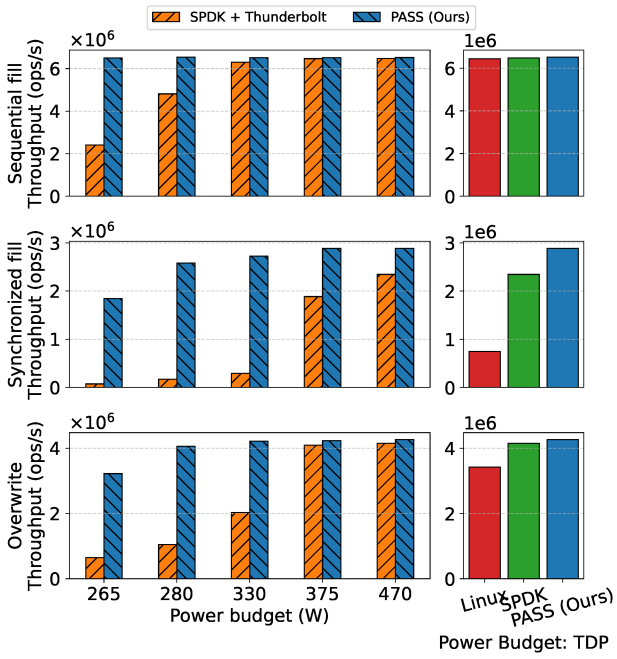


Figure 9. Throughput of db_bench benchmarks

writes to a new database), fillsync (synchronized writes requiring high durability), and overwrite (updates to existing data). Each benchmark operates on a database with 4 million entries. The results in Figure 9 highlight PASS's superior performance, regardless of power constraints. For fillseq, PASS achieves up to 2.71× the throughput of Thunderbolt at 265W. In fillsync, where durability demands are stringent, PASS outperforms Linux by 2.87× and SPDK dynamic scheduler by 23.1% at TDP, while it outperforms Thunderbolt with 14.95× the throughput at 280W. Similar trends are observed in overwrite, where PASS consistently delivers better performance compared to baseline systems with 24.7% higher throughput than Linux at TDP and up to 2.88× higher throughput than SPDK+Thunderbolt at 280W power budget.

Key-value store (YCSB). The YCSB benchmarks, run on RocksDB, provide insight into PASS's performance under common key-value store access patterns. Workloads A through D and F, covering read-heavy, write-heavy, and mixed scenarios, are evaluated on a database with 100 million records and 200 million operations. Workload E is excluded as it focuses on small-scale updates that do not stress storage.

Figure 10 shows that PASS achieves higher performance than baselines. At TDP, PASS achieves 19.2%, 15.9%, 6.3%, 6.4%, 10.1% lower p99 tail latency than Linux and 9.1%, 3.4%, 1.2%, 6.1%, 5.5% lower than SPDK dynamic. Under power cap, for read-intensive workloads such as Workload C, PASS achieves 432.35× lower p99 tail latency than Thunderbolt at

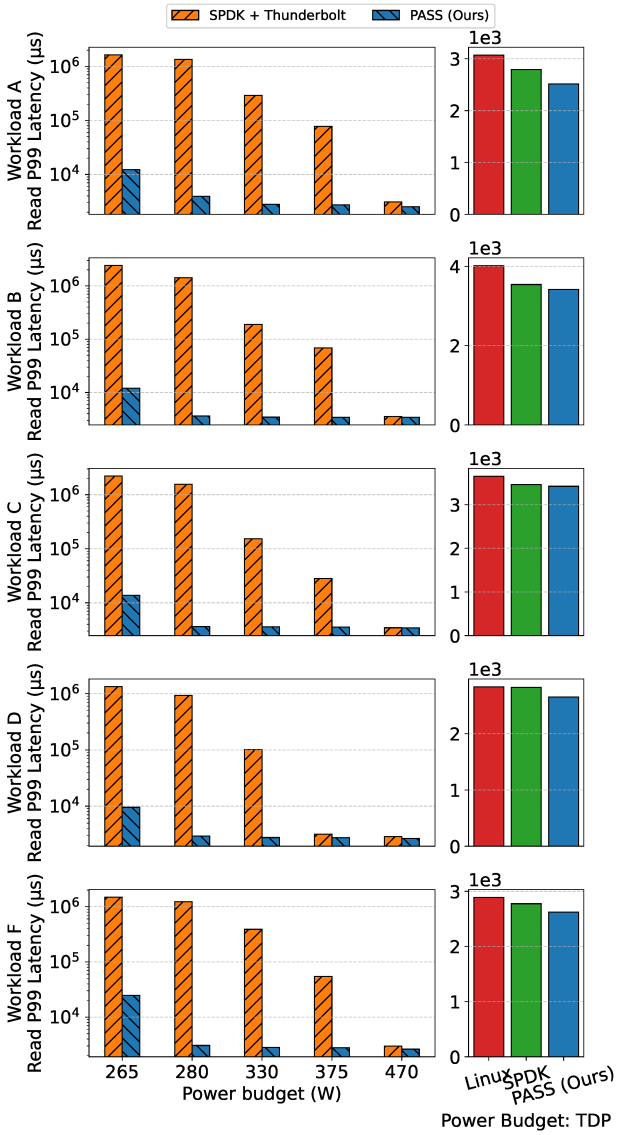


Figure 10. Average P99 latency of read in YCSB workloads.

280W. In mixed workloads like Workload B, PASS provides 391.95× smaller tail latency than Thunderbolt at 280W.

Summary. Overall, these application benchmarks confirm that PASS effectively maximizes performance under power budgets across a diverse set of workloads by dynamically adapting its resource usage.

5.3 Energy Efficiency

Although not directly targeting energy reduction, PASS often achieves better energy efficiency than competing techniques due to its fine-grained and stable power reallocation between CPU and SSD, guided by offline profiling and online feedback. Energy per operation can be derived from our reported

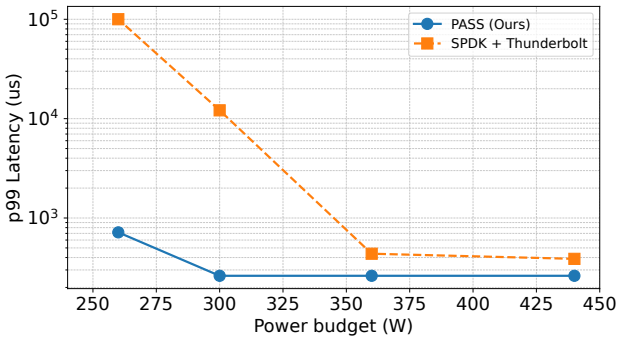


Figure 11. Performance of latency-critical job in a mix with throughput-intensive background load.

results in Figure 8. For example, at a budget of 300W on the Varmail workload, PASS (299.134 uJ/op) achieves 6.3% better energy efficiency than Linux (319.896 uJ/op) and is over 12× more energy efficient than SPDK+Thunderbolt (3655.772 uJ/op). For Webserver at 300W, PASS (653.386 uJ/op) operates within 4% of the energy efficiency of Linux (625.951 uJ/op). At 350W, PASS (666.717 uJ/op) is over 5× more energy efficient than SPDK+Thunderbolt (3751.3 uJ/op). These results show that PASS achieves strong energy efficiency at constrained budgets—its intended use case.

Unconstrained scenarios are also possible operating points. For example, during certain periods, power may be free or overprovisioned (e.g., due to excess renewables). In such unconstrained scenarios (e.g., 500W in Figure 8), PASS (722.506 uJ/op) consumes about 20% more energy per operation than Linux (592.008 uJ/op) and 9.3% more than SPDK+Thunderbolt (660.818 uJ/op). This is expected since performance saturates at high power. It is also acceptable in such overprovisioned scenarios.

5.4 Performance Isolation

To evaluate how well PASS can isolate the performance of workloads while operating under power budget, we evaluate a scenario with a mixture of latency-sensitive and throughput-oriented clients. Specifically, our scenario comprises: (1) a latency-critical workload with 4 KiB synchronous random reads with 1 job, and (2) a background workload with 128 KiB sequential writes on 9 additional jobs. This setup reflects a realistic scenario where data centers must maintain the performance of critical applications (e.g., transactional systems) while reducing power consumption for less critical background tasks.

The results are shown in Figure 11. Without any power cap, PASS achieves 32.5% less 99th percentile latency than baselines using SPDK default dynamic scheduler, as PASS uses more cores and separates the critical from background load. This performance separation is more significant when the system is power capped. At a 300W power budget, significant differences emerge: PASS enforces a disk bandwidth control of 10 MiB/s for the background workloads, effectively isolating the latency-critical workload. With this adjustment, the 4 KiB random read workload achieves performance metrics comparable to the high-budget scenario. By contrast, the baselines struggle under the same 300W power budget: RAPL's tail latency increased to 742 μs and Thunderbolt's increased to 12ms while PASS stays at 262 μs, 64.7% and 97.8% lower than each baseline, respectively. At a power budget of 260W, although PASS's tail latency also increases by 2.74× to 717 μs due to the application of CPU bandwidth control, SPDK+Thunderbolt exercises CPU bandwidth control much more aggressively. PASS achieves about 140× lower 99th percentile latency than SPDK+Thunderbolt.

These results highlight PASS's ability to reduce interference from background workloads and preserve critical workload performance under strict power budgets. PASS achieves this via its cross-layer power management, which employs fine-grained CPU attribution control and differential disk bandwidth control, allowing it to prioritize critical workloads even during power-constrained conditions.

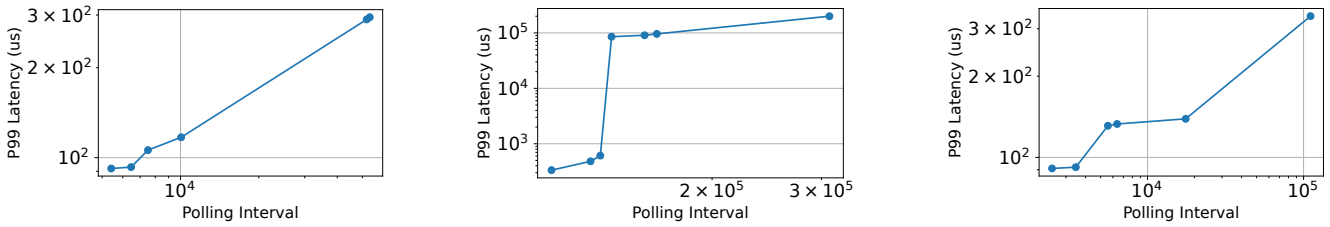
5.5 Cross-Platform Validation

We validate that PASS generalizes across different server platforms, demonstrating that power control policies must be tailored to each system's characteristics. To this end, we use PASS profiling results from two servers: our in-house testbed (Server A) and a CloudLab [9] c6620 server (Server B) equipped with an Intel Xeon Gold 5512U processor. For each server, we select the CPU configuration that achieves approximately 50% of the processor's thermal design power (TDP), denoted as Configuration A (from Server A) and Configuration B (from Server B), respectively.

We conduct a cross-validation study by running a 4 KiB random write workload using both configurations on both servers. To ensure fair comparison, we scale the configurations according to each server's CPU specifications—specifically, the number of cores and the TDP. Table 1 summarizes the results. Each configuration performs the best on the server it was profiled for, validating that PASS successfully generalizes across platforms.

Table 1. P99 latency (in microseconds) of 4 KiB random write workloads using cross-platform CPU configurations. Configuration A and B correspond to the best 50%-max CPU power configurations for Server A and B, respectively.

	Configuration A	Configuration B
Server A	**400**	611
Server B	76	**52**



(a) RAPL power limits. **(b)** CPU bandwidth control. **(c)** CPU core jailing.

Figure 12. IO polling interval and 99th percentile latency of 4 KiB random write under different CPU power control mechanisms.

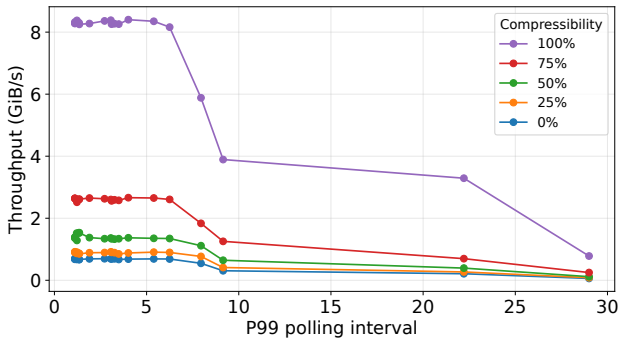


Figure 13. IO throughput across CPU power control mechanisms and levels of compressibility.

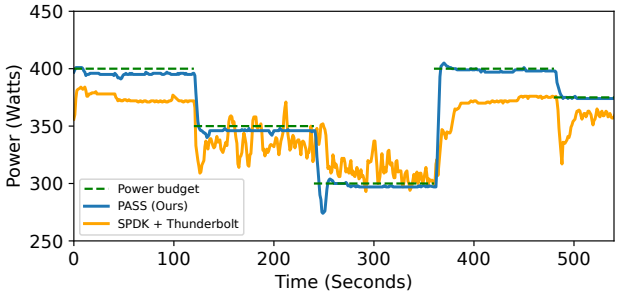


Figure 14. 64 KiB random read, varying power budget.

5.6 IO Polling Interval as Performance Proxy

To evaluate whether the IO polling interval is a suitable proxy for overall system performance in PASS, we examine its correlation with VM storage latency. Specifically, we analyze the relationship between the 99th percentile polling interval, as measured from SPDK pollers, and the end-to-end 99th percentile latency observed by VMs under a 16-job, 4 KiB random write workload.

We vary CPU power control settings using three mechanisms that impact the duration of the IO polling interval:

- **CPU core count:** 1, 2, 3, 4, 6, and 8 cores
- **RAPL power limits:** 100W (minimum achievable), 110W, 120W, 130W, 140W, and 160W
- **CPU bandwidth caps:** 1%, 5%, 10%, 15%, 25%, and 50%

Figure 12 shows that, across all control mechanisms, there is a strong monotonic relationship: as the polling interval increases, so does 99th percentile workload latency. This trend also holds for system throughput.

To validate the generality of the IO polling interval as a performance proxy beyond IO-bound workloads, we examine a typical CPU-bound storage task: compression. In particular, we evaluate the relationship between 99th percentile polling interval and IO throughput across different CPU power control mechanisms and levels of data compressibility. Data compressibility varies CPU to IO intensity. As shown in Figure 13, across all power control mechanisms and compressibility levels, there is a monotonic relationship: as the IO polling interval increases due to CPU power control, the throughput of compression decreases. These observations validate that the IO polling interval can serve as a reliable proxy for system performance, particularly for evaluating the relative impact of different CPU configurations.

5.7 Power Control Stability and Performance

We evaluate how PASS and SPDK+Thunderbolt respond to dynamically changing power budgets. In this experiment, a sustained 64 KiB random read workload is issued from the client. The power budget begins at 400 W and is then changed according to Figure 14. The experiment runs for a total of 10 minutes, capturing steady-state behavior across dynamic budget transitions.

As shown in Figure 14, PASS adapts quickly and accurately to each budget transition, maintaining system power consumption within 2% of the budget. PASS's maximum continuous power budget violation is for 5 seconds, with a magnitude of 1.5%. We can see that PASS undershoots each budget initially when reducing power and overshoots when increasing power. However, in both cases, PASS can correct within 5 seconds, showing the effect of PASS's two-stage proportional power control. For the power capping down to 300 W, PASS incorporates SSD throttling, which leads to a slightly longer period of underpower than that for 350 W. In contrast, the baseline system exhibits high variability. At the

lowest budget level of 300 W, Thunderbolt also repeatedly violates the power cap due to CPU bandwidth being reduced to its minimum value of 0.2%, which impairs Thunderbolt's ability to maintain system power effectively.

6 Related Work

Server power management systems. Prior work on server power management has largely focused on protecting power infrastructure from overload. Such systems typically throttle processors via RAPL or CPU bandwidth control to keep power consumption within server, rack, or data center-wide limits [25–28, 37, 53, 58].

Beyond node-local CPU controls, several efforts have explored cross-layer and multi-node power coordination. For example, Raghavendra, *et al.* [36] propose a hierarchical scheme spanning VMs, servers, and clusters, assuming perfectly power-proportional servers. Meisner, *et al.* [32] coordinate CPUs, DRAM, and HDD power management for energy savings under low utilization, while PowerNap [31] transitions servers between high and low-power states to exploit idle periods. Finally, BlinkFS [40] addresses large-scale variability from renewable energy or utility disruptions by coordinating file systems with external power availability by powering up and down servers in a cluster.

PASS complements these approaches by performing dynamic, bottleneck-aware power reallocation between CPUs and SSDs, guided by offline profiling and online feedback. This enables fine-grained control across the power-performance curve, as opposed to coarse-grained, binary, or server-level schemes. This allows PASS to achieve higher performance and a broader dynamic range than prior techniques on server power management.

Storage power proportionality and efficiency. Prior work explored power proportionality in ensembles of hard disk-based storage servers [48, 51, 52]. Because HDDs lack power proportionality, these systems employ specialized replication schemes to power down some disks while maintaining acceptable availability. Khatib *et al.* [23] propose I/O queue resizing to limit HDD power consumption, but their approach offers limited power scaling, as it focuses solely on HDDs without addressing cross-layer opportunities to reduce total server power. Recent studies address SSD power efficiency [5, 15], aiming to reduce SSD power draw. In contrast, PASS advocates for power adaptivity, rather than proportionality or efficiency, and is designed to meet strict server-level power budgets and leverages the available budget to improve performance.

Storage system power studies. The power and energy characteristics of storage systems have received considerable attention [6, 13, 14, 30, 34, 44, 50, 56]. For instance, Cao *et al.* [5] analyze how CPU power control mechanisms affect storage system performance, while Xie *et al.* [54] develop a power-performance model across different power control strategies. These efforts provide valuable insights, and their SSD characterization techniques can be integrated into our SSD power model. Our work complements these studies by focusing on the broader system-level relationship between power management and storage performance, offering new opportunities for cross-layer optimization.

7 Conclusion

PASS is a power-adaptive storage system that maximizes performance within a given power budget, as well as maximizing power dynamic range. PASS integrates control mechanisms across CPU and SSD layers to dynamically adjust resource utilization in response to changing power constraints. PASS outperforms existing capped power-proportional systems, achieving broader power adaptivity, higher throughput, and lower latency across diverse workloads. By prioritizing power adaptivity, PASS offers a robust solution to the growing power management challenges in modern data centers.

Acknowledgments

We thank our shepherd Christopher Stewart and the anonymous reviewers for their valuable feedback on various aspects of the paper. This work is supported by NSF grants 2104548, 2148209, and 2212193, as well as the University of Washington Center for the Future of Cloud Infrastructure (FOCI). This work was also generously supported by the PRISM Research Center, a JUMP Center cosponsored by SRC and DARPA.

References

[1] Sebastian Angel, Mihir Nanavati, and Siddhartha Sen. Disaggregation and the application. In *12th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 20)*, 2020.

[2] Jens Axboe. fio - Flexible I/O tester rev. 3.36. https://fio.readthedocs.io/en/latest/fio_doc.html, 2017.

[3] David Britton, Simone Campana, and Bernd Panzer-Stradel. A holistic study of the WLCG energy needs for the LHC scientific program. In *EPJ Web of Conferences*, volume 295, 2024.

[4] Aaron Call, Jordà Polo, and David Carrera. Workload-aware placement strategies to leverage disaggregated resources in the datacenter. *IEEE Systems Journal*, 16(1):1697–1708, 2021.

[5] Hankun Cao, Shai Bergman, Si Sun, Yanbo Albert Zhou, Xijun Li, Jun Gao, Zhuo Cheng, and Ji Zhang. Answering the Call to ARMs with PACER: Power-Efficiency in Storage Servers. In *The International Conference on Massive Storage Systems and Technology*, 2024.

[6] Seokhei Cho, Changhyun Park, Youjip Won, Sooyong Kang, Jaehyuk Cha, Sungroh Yoon, and Jongmoo Choi. Design tradeoffs of SSDs: From energy consumption's perspective. *ACM Transactions on Storage (TOS)*, 11(2):1–24, 2015.

[7] Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. Benchmarking cloud serving systems with YCSB. In *Proceedings of the 1st ACM Symposium on Cloud Computing 2010 (SoCC'10)*, page 143–154, New York, NY, USA, 2010. Association for Computing Machinery (ACM).

[8] Jesse David Dinneen and Ba Xuan Nguyen. How Big Are Peoples' Computer Files? File Size Distributions Among User-managed Collections. *arXiv preprint arXiv:2107.03272*, 2021.

[9] Dmitry Duplyakin, Robert Ricci, Aleksander Maricq, Gary Wong, Jonathon Duerig, Eric Eide, Leigh Stoller, Mike Hibler, David Johnson, Kirk Webb, Aditya Akella, Kuangching Wang, Glenn Ricart, Larry Landweber, Chip Elliott, Michael Zink, Emmanuel Cecchet, Snigdhaswin Kar, and Prabodh Mishra. The design and operation of CloudLab. In *Proceedings of the USENIX Annual Technical Conference (ATC)*, pages 1–14, July 2019.

[10] NVM Express. NVM Express Base Specification, Revision 2.1. https://nvmexpress.org/specification/nvm-express-base-specification/, 2024.

[11] Laurence Goasduff. How to Make the Data Center Eco-Friendly: Q&A with Philip Dawson. https://www.gartner.com/en/newsroom/press-releases/2022-12-06-how-to-make-the-data-center-eco-friendly, 2022.

[12] Jonmichael Hands, Dennis Worley, and Lakhveer Kaur. Technology power features - NVM express. https://nvmexpress.org/resource/technology-power-features/, 2022.

[13] Bryan Harris and Nihat Altiparmak. Ultra-Low Latency SSDs' Impact on Overall Energy Efficiency. In *Proceedings of the 12th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage'20)*, Berkeley, CA, USA, 2020. USENIX Association.

[14] Bryan Harris and Nihat Altiparmak. When poll is more energy efficient than interrupt. In *Proceedings of the 14th ACM Workshop on Hot Topics in Storage and File Systems (HotStorage '22)*, New York, NY, USA, 2022. Association for Computing Machinery (ACM).

[15] Satoshi Imamura, Eiji Yoshida, and Kazuichi Oe. Reducing CPU Power Consumption with Device Utilization-Aware DVFS for Low-Latency SSDs. *IEICE TRANSACTIONS on Information and Systems*, 102(9):1740–1749, 2019.

[16] KIOXIA America Inc. EDSFF - A New SSD Form Factor for Next Gen Servers and Storage. https://americas.kioxia.com/en-us/business/ssd/solution/edsff.html, 2025.

[17] Intel. DCMI: Data Center Manageability Interface Specification v1.5 Revision 1.0. https://www.intel.com/content/dam/www/public/us/en/documents/technical-specifications/dcmi-v1-5-rev-spec.pdf, 2011.

[18] Intel. Intel® 64 and IA-32 Architectures Software Developer's Manual Combined Volumes 3A, 3B, 3C, and 3D: System Programming Guide. https://www.intel.com/content/www/us/en/content-details/843836/intel-64-and-ia-32-architectures-software-developer-s-manual-combined-volumes-3a-3b-3c-and-3d-system-programming-guide.html?wapkw=Running%20Average%20Power%20Limit, 2024.

[19] Linux kernel. The Linux-native asynchronous I/O facility (aio) library. https://pagure.io/libaio, 2022.

[20] Linux kernel. ext4 General Information. https://docs.kernel.org/admin-guide/ext4.html, 2025.

[21] Linux kernel development community. CFS Bandwidth Control. https://docs.kernel.org/scheduler/sched-bwc.html, 2024.

[22] Linux kernel development community. Power Capping Framework. https://docs.kernel.org/power/powercap/powercap.html, 2024.

[23] Mohammed G Khatib and Zvonimir Bandic. {PCAP}: Performance-aware power capping for the disk drive in the cloud. In *14th USENIX Conference on File and Storage Technologies (FAST 16)*, pages 227–240, 2016.

[24] Ana Klimovic, Christos Kozyrakis, Eno Thereska, Binu John, and Sanjeev Kumar. Flash storage disaggregation. In *Proceedings of the 10th European Conference on Computer Systems (EuroSys'16)*, New York, NY, USA, 2016. Association for Computing Machinery (ACM).

[25] Alok Gautam Kumbhare, Reza Azimi, Ioannis Manousakis, Anand Bonde, Felipe Frujeri, Nithish Mahalingam, Pulkit A Misra, Seyyed Ahmad Javadi, Bianca Schroeder, Marcus Fontoura, et al. Prediction-Based power oversubscription in cloud platforms. In *2021 USENIX Annual Technical Conference (USENIX ATC'21)*, Berkeley, CA, USA, 2021. USENIX Association.

[26] Charles Lefurgy, Xiaorui Wang, and Malcolm Ware. Server-level power control. In *Fourth International Conference on Autonomic Computing (ICAC'07)*, pages 4–4. IEEE, 2007.

[27] Shaohong Li, Xi Wang, Faria Kalim, Xiao Zhang, Sangeetha Abdu Jyothi, Karan Grover, Vasileios Kontorinis, Nina Narodytska, Owolabi Legunsen, Sreekumar Kodakara, et al. Thunderbolt: Throughput-Optimized, Quality-of-Service-Aware Power Capping at Scale. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI'20)*, Berkeley, CA, USA, 2020. USENIX Association.

[28] Yang Li, Charles R. Lefurgy, Karthick Rajamani, Malcolm S. Allen-Ware, Guillermo J. Silva, Daniel D. Heimsoth, Saugata Ghose, and Onur Mutlu. A scalable priority-aware approach to managing data center server power. In *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 701–714, 2019.

[29] Liuzixuan Lin and Andrew A Chien. Adapting Datacenter Capacity for Greener Datacenters and Grid. In *Proceedings of the 14th ACM International Conference on Future Energy Systems*, page 200–213, 2023.

[30] Sara Mcallister, Fiodar Kazhamiaka, Daniel S. Berger, Rodrigo Fonseca, Kali Frost, Aaron Ogus, Maneesh Sah, Ricardo Bianchini, George Amvrosiadis, Nathan Beckmann, and Gregory R. Ganger. A Call for Research on Storage Emissions. *SIGENERGY Energy Inform. Rev.*, 4(5):67–75, April 2025.

[31] David Meisner, Brian T. Gold, and Thomas F. Wenisch. PowerNap: eliminating server idle power. *SIGARCH Comput. Archit. News*, 37(1):205–216, March 2009.

[32] David Meisner, Christopher M. Sadler, Luiz André Barroso, Wolf-Dietrich Weber, and Thomas F. Wenisch. Power management of online data-intensive services. In *Proceedings of the 38th Annual International Symposium on Computer Architecture*, ISCA '11, page 319–330, New York, NY, USA, 2011. Association for Computing Machinery.

[33] Jaehong Min, Ming Liu, Tapan Chugh, Chenxingyu Zhao, Andrew Wei, In Hwan Doh, and Arvind Krishnamurthy. Gimbal: enabling multi-tenant storage disaggregation on SmartNIC JBOFs. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, page 106–122, 2021.

[34] Jinha Park, Sungjoo Yoo, Sunggu Lee, and Chanik Park. Power modeling of solid state disk for dynamic power management policy design in embedded systems. In *Software Technologies for Embedded and Ubiquitous Systems: 7th IFIP WG 10.2 International Workshop*, pages 24–35. Springer, 2009.

[35] Jonggyu Park, Theano Stavrinos, Simon Peter, and Thomas Anderson. Empower: The case for a cloud power control plane. *ACM SIGENERGY Energy Informatics Review*, 4(5):76–83, 2024.

[36] Ramya Raghavendra, Parthasarathy Ranganathan, Vanish Talwar, Zhikui Wang, and Xiaoyun Zhu. No "power" struggles: coordinated multi-level power management for the data center. *SIGOPS Oper. Syst. Rev.*, 42(2):48–59, March 2008.

[37] Parthasarathy Ranganathan, Phil Leech, David Irwin, and Jeffrey Chase. Ensemble-level power management for dense blade servers. *ACM SIGARCH computer architecture news*, 34(2):66–77, 2006.

[38] Varsha Rao and Andrew A. Chien. Understanding the Operational Carbon Footprint of Storage Reliability and Management. *SIGENERGY Energy Inform. Rev.*, 4(5):180–187, April 2025.

[39] Samsung. Samsung PM1743. https://semiconductor.samsung.com/us/ssd/enterprise-ssd/pm1743/, 2024.

[40] Navin Sharma, David Irwin, and Prashant Shenoy. BlinkFS: A distributed file system for intermittent power. *Sustainable Computing: Informatics and Systems*, 6:69–80, 2015.

[41] SNIA. SSD Form Factors: EDSFF. https://www.snia.org/forums/cmsi/knowledge/formfactors#edsff, 2025.

[42] Solidigm. Solidigm D7-P5510. https://www.solidigm.com/products/data-center/d7/p5510.html, 2025.

[43] SPDK. Storage Performance Development Kit. https://spdk.io, 2024.

[44] Sidharth Sundar, William Simpson, Jacob Higdon, Caeden Whitaker, Bryan Harris, and Nihat Altiparmak. Energy Implications of IO Interface Design Choices. In *Proceedings of the 15th ACM Workshop on Hot Topics in Storage and File Systems (HotStorage '23)*, New York, NY,

USA, 2023. Association for Computing Machinery (ACM).

[45] Vasily Tarasov, Erez Zadok, and Spencer Shepler. Filebench: A Flexible Framework for File System Benchmarking. *login Usenix Mag.*, 41, 2016.

[46] Facebook Database Engineering Team. RocksDB. https://rocksdb.org/, 2024.

[47] Facebook Database Engineering Team. RocksDB Benchmarking Tools. https://github.com/facebook/rocksdb/wiki/Benchmarking-tools, 2024.

[48] Eno Thereska, Austin Donnelly, and Dushyanth Narayanan. Sierra: practical power-proportionality for data center storage. In *Proceedings of the 6th European Conference on Computer Systems (EuroSys'11)*, New York, NY, USA, 2011. Association for Computing Machinery (ACM).

[49] Muhammad Tirmazi, Adam Barker, Nan Deng, Md E Haque, Zhijing Gene Qin, Steven Hand, Mor Harchol-Balter, and John Wilkes. Borg: the next generation. In *Proceedings of the 15th European Conference on Computer Systems (EuroSys'20)*, pages 1–14, New York, NY, USA, 2020. Association for Computing Machinery (ACM).

[50] Erica Tomes and Nihat Altiparmak. A comparative study of HDD and SSD RAIDs' impact on server energy consumption. In *Proceedings of the 2017 IEEE International Conference on Cluster Computing (CLUSTER '17)*, Washington, DC, USA, 2017. IEEE Computer Society.

[51] Akshat Verma, Ricardo Koller, Luis Useche, and Raju Rangaswami. SRCMap: Energy Proportional Storage Using Dynamic Consolidation. In *Proceedings of the 8th USENIX Conference on File and Storage Technologies (FAST'10)*, Berkeley, CA, USA, 2010. USENIX Association.

[52] Charles Weddle, Mathew Oldham, Jin Qian, An-I Andy Wang, Peter Reiher, and Geoff Kuenning. PARAID: A gear-shifting power-aware RAID. *ACM Transactions on Storage (TOS)*, 3(3):13–es, 2007.

[53] Qiang Wu, Qingyuan Deng, Lakshmi Ganesh, Chang-Hong Hsu, Yun Jin, Sanjeev Kumar, Bin Li, Justin Meza, and Yee Jiun Song. Dynamo: Facebook's data center-wide power management system. *ACM SIGARCH Computer Architecture News*, 44(3):469–480, 2016.

[54] Dedong Xie, Theano Stavrinos, Kan Zhu, Simon Peter, Baris Kasikci, and Thomas Anderson. Can Storage Devices be Power Adaptive? In *Proceedings of the 16th ACM Workshop on Hot Topics in Storage and File Systems (HotStorage'24)*, pages 47–54, New York, NY, USA, 2024. Association for Computing Machinery (ACM).

[55] Zhengyu Yang, Manu Awasthi, Mrinmoy Ghosh, Janki Bhimani, and Ningfang Mi. I/O Workload Management for All-Flash Datacenter Storage Systems Based on Total Cost of Ownership. *IEEE Transactions on Big Data*, 8:332–345, 2018.

[56] Balgeun Yoo, Youjip Won, Seokhei Cho, Sooyong Kang, Jongmoo Choi, and Sungroh Yoon. SSD Characterization: From Energy Consumption's Perspective. In *Proceedings of the 3rd USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage'11)*, Berkeley, CA, USA, 2011. USENIX Association.

[57] Shaoxun Zeng, Xiaojian Liao, Hao Guo, and Youyou Lu. Volley: Accelerating Write-Read Orders in Disaggregated Storage. In *Proceedings of the 19th European Conference on Computer Systems (EuroSys'24)*, pages 657–673, New York, NY, USA, 2024. Association for Computing Machinery (ACM).

[58] Chaojie Zhang, Alok Kumbhare, Ioannis Manousakis, Deli Zhang, Pulkit Misra, Rod Assis, Kyle Woolcock, Nithish Mahalingam, Brijesh Warrier, David Gauthier, Lalu Kunnath, Steve Solomon, Osvaldo Morales, Marcus Fontoura, and Ricardo Bianchini. Flex: High-Availability Datacenters With Zero Reserved Power. In *Proceedings of the 48th Annual International Symposium on Computer Architecture (ISCA'22)*, New York, NY, USA, 2021. Association for Computing Machinery (ACM).

[59] Mark Zhao, Satadru Pan, Niket Agarwal, Zhaoduo Wen, David Xu, Anand Natarajan, Pavan Kumar, Ritesh Tijoriwala, Karan Asher, Hao Wu, et al. Tectonic-Shift: A Composite Storage Fabric for Large-Scale ML Training. In *2023 USENIX Annual Technical Conference (USENIX ATC'23)*, Berkeley, CA, USA, 2023. USENIX Association.