Answering the Call to ARMs with PACER: Power-Efficiency in Storage Servers

Hankun Cao*
Huawei Technologies Co.
caohankun1@huawei.com

Shai Bergman*
Huawei Zurich Research Center
shai.aviram.bergman@huawei.com

Si Sun
Huawei Technologies Co.
sunsi10@huawei.com

Yanbo Albert Zhou
University of Warwick
albert.zhou@warwick.ac.uk

Xijun Li
Huawei Technologies Co.
xijun.li@huawei.com

Jun Gao
Huawei Technologies Co.
gaojun@huawei.com

Zhuo Cheng
Huawei Zurich Research Center
chengzhuo@huawei.com

Ji Zhang†
Huawei Zurich Research Center
dr.jizhang@huawei.com

***Abstract*—The escalating power consumption in data-centers poses a critical challenge, particularly with the surging computational demand. While current research emphasizes AI and accelerator power consumption, there exists a notable gap in efforts to optimize on-premise storage server power usage, which accounts for 25-30% of the power consumption in the data-center.**

In this paper, we introduce *PACER*, a power-saving subsystem for ARM-based, all-flash storage servers. PACER's design is based on a comprehensive analysis of the utilization and power consumption of storage servers in data-centers. Unlike conventional power-saving approaches, PACER is guided by storage server-specific IO metrics, employs advanced monitoring to assess the server's operational state, and predicts the potential IO performance impact of power-saving techniques. It navigates the trade-off between performance and power-efficiency by adhering to configurable performance tolerance boundaries. Our contributions include a thorough analysis of data-center storage servers, an efficient core resource manager for ARM-based storage servers, and an IO-centric power-saving mechanism.

We extensively evaluate PACER with synthetic and real-world workloads. In the evaluation of real-world traces, PACER achieves a 1.23× higher IOPS/watt over the native system implementation. In comparison, Linux's on-demand governor improves the IOPS/watt by a factor of 1.02×, while experiencing higher average and tail IO latencies than observed in PACER. In the MLPerf Storage benchmark for 3D-UNET and BERT, PACER achieves power-saving gains of 1.15× and 1.28× over the native system implementation, respectively, without incurring any performance degradation.

***Index Terms*—Storage servers, Power efficiency, SSD**

I. Introduction

Escalating computational demands have precipitated a marked rise in the energy consumption of data-centers, elevating concerns regarding power efficiency [1]–[5]. As digital services and applications proliferate, the imperative to optimize data-center energy utilization intensifies.

The surge in artificial intelligence and data analytics applications necessitates substantial enhancements in data-center storage capabilities [6], [7] and imposes rigorous performance requirements on storage servers [8], [9]. These infrastructures are concurrently pressured to scale both storage capacity and processing throughput [10], [11], as underscored by their increasing usage [12]–[14].

Data-centers are increasingly adopting all-flash storage servers [15]–[17], as the cost of flash-based storage media, such as QLC NAND, continues to decline [18]. These servers deliver superior performance relative to their HDD-based counterparts [19], [20], addressing the escalating demand for performance in emerging data-center workloads.

Storage servers account for approximately 25% to 30% of the total power consumption within data-centers [21], [22]. Current research initiatives focus on enhancing the power consumption of artificial intelligence and other accelerators commonly deployed in data-centers [23]–[25]. However, there is a notable gap in recent efforts to optimize the power consumption of storage servers, particularly in light of the emergence of cost-effective flash storage and advancements in performant and power-efficient CPU architectures.

Current power-saving solutions are broad in scope and do not address the unique requirements of storage servers. These approaches do not consider the potential influence of power-saving methods on a storage server's IO throughput and latency, and they are unaware of the utilization and access patterns specific to storage servers. Therefore, these methods are constrained in their capacity to efficiently conserve power and may compromise storage system performance. For instance, the Linux on-demand power governor [26], while adjusting the CPU core frequency based on CPU utilization, fails to consider its impact on IO performance, and lacks other power-saving methods such as power-gating.

To further the power-saving efforts in data-centers, prior works investigate the feasibility of replacing x86-based CPUs with energy-efficient ARM-based CPUs for compute servers [27]–[30]. However, the comprehensive examination of leveraging server-grade ARM CPUs in enterprise storage servers to achieve the goal of mitigating power consumption in storage systems, has not been extensively conducted. This is particularly relevant given the substantial computational demands that are undertaken by storage servers [31], [32].

In this work, we introduce **PACER**, a power-saving subsystem designed for all-flash storage servers. While conventional

*Equal contribution
†Ji Zhang is the corresponding author

power-saving approaches typically account for application-agnostic metrics such as CPU utilization [26], PACER is guided by IO metrics specific to storage servers. It employs advanced, continuous monitoring methods to assess the operating condition of the storage server and anticipates the potential IO performance implications of applying power-saving techniques like power gating, and dynamic voltage and frequency scaling. It adheres to configurable performance boundaries, facilitating the navigation of the trade-off between performance and power-efficiency.

PACER systematically optimizes power conservation opportunities within data-center storage servers, specifically focusing on the most power-intensive hardware components. This optimization is realized through an in-depth analysis conducted in this study, examining storage server utilization across numerous data-centers.

To further enhance power efficiency, PACER is designed for efficient operation on server-class ARM-based CPUs. It utilizes specialized task-scheduling methods to effectively fulfill the performance criteria of data-center storage servers, and compete with its x86 counterparts.

To realize PACER, we make the following contributions:

(1) Thorough analysis of data-center storage servers. We thoroughly analyze the utilization patterns and power consumption of storage servers deployed across 80 data-centers, and identify key opportunities for power-saving. We also assess the impact of conventional power-saving techniques, such as power gating and dynamic voltage frequency scaling, on the IO performance and overall power-consumption of storage servers.

(2) Task manager for ARM-based storage servers. We design a core resource manager and task scheduler optimized for ARM architectures. The resource manager harnesses the abundant cores integrated into server-grade ARM CPUs.

(3) Power-saving subsystem for storage servers. An IO-centric power-saving subsystem designed for all-flash storage servers. Conventional power-saving approaches overlook their impact on IO performance in storage servers, focusing solely on general system metrics such as CPU utilization. PACER's feedback-driven subsystem continuously monitors IO performance, anticipates the impact of power-saving methods on IO performance, and maintains a pre-defined Service Level Agreement (SLA) based on configurable hyperparameters.

PACER is implemented on a commercially available enterprise-grade storage server equipped with a server-grade ARM CPU and an all-flash array of SSDs.

We evaluate PACER using synthetic and real-world traces obtained from numerous data-centers, as well as MLPerf Storage benchmarks [33]. Our results show that PACER achieves significant power-savings while upholding its pre-defined SLA. Under a pre-defined IO latency tolerance factor of 1.2×, PACER demonstrates a 1.23× improvement in IOPS/WATT over a native system implementation in a real-world trace, taking into account the overall power consumption of the storage server. It incurs a 1.19× increase in average IO latency, which is within the predefined IO latency tolerance threshold. In comparison, the Linux on-demand governor achieves a modest 1.02× improvement in IOPS/WATT and incurs an average IO latency increase of 1.26× over a native system implementation. For MLPerf Storage benchmark [33], PACER achieves significant power-savings, consuming 1.15× and 1.28× less power than the native system implementation for 3D-UNET and BERT, respectively. Importantly, these power-saving achievements are attained without incurring performance degradation in the benchmarks.

II. Background

Storage servers. Storage servers allow for the separation of storage and compute resources, which enhances data-center scalability, flexibility and cost-efficiency [34]–[36]. To ensure high reliability and performance, storage servers execute multiple operations in the critical path of IO requests, while lower priority operations are performed as background jobs. For example, storage servers perform caching, compression, indexing, and data integrity checks in the critical path, while tasks such as file-system garbage-collection and telemetry are done in the background. Fig. 1 depicts a typical high-level architecture of storage servers. The IO data path consists of five major layers.

The *protocols layer* handles requests of various protocols such as Storage Area Network (SAN) or Network-Attached Storage (NAS), and converts them into a unified request format. It also provides services such as user management, client authentication, and Access Control List (ACL). The request is forwarded to the *data services layer*, which is responsible for managing the file systems in the storage server. The *global cache layer* attempts to fulfill IO requests from the storage system's cache. The cache is typically composed of DRAM, and leverages fast storage media like Storage Class Memory (SCM) for persistence. The *space manager layer* abstracts the discrete physical storage resources' address spaces into a contiguous logical address space to the upper layers. The layer also translates the request's logical address to its physical location. The bottom layer is the *storage pool layer*, which manages all the physical storage resources in the storage server. This layer also implements RAID, erasure coding, data reconstruction during failures, file-system garbage-collection, and other physical media management functionalities.

Storage systems optimize for performance by implementing several mechanisms such as caching and prefetching. Despite such optimizations, attaining optimal performance for end-users may be impeded by CPU bottlenecks in the system [31], [36]; numerous storage server tasks across various layers necessitate substantial computing resource [31]. In the absence of sufficient computational capabilities, inefficiencies within the system may manifest, thereby impacting the overall system performance.

Power efficiency and performance of ARM and x86. ARM and x86 CPUs operate on distinct Instruction Set Architectures (ISAs). ARM CPUs employ the Reduced Instruction Set Computing (RISC), while x86 CPUs employ Complex Instruction

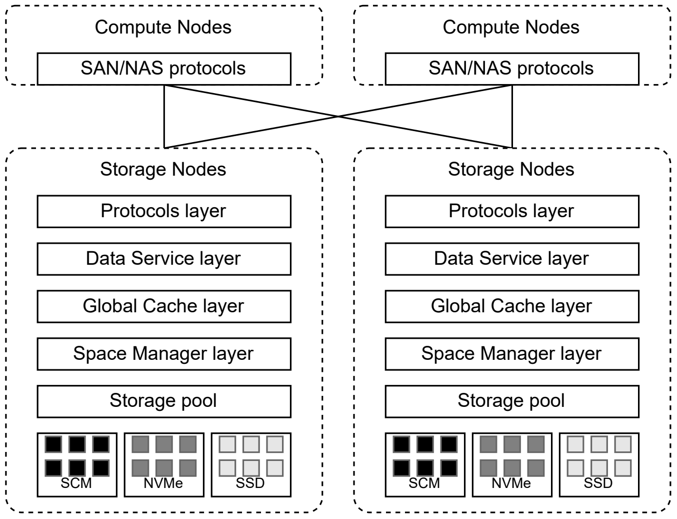

Fig. 1: High-level architecture of typical storage servers

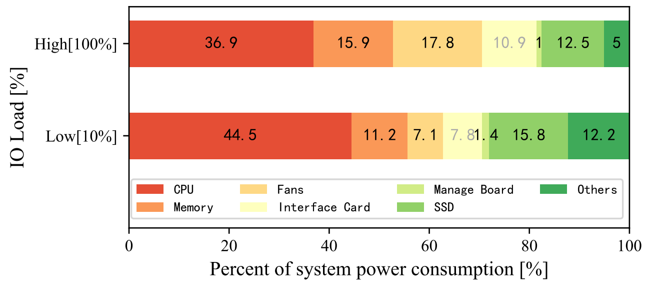

Fig. 2: Storage server power consumption per component under typical-low (10%) and high (100%) IO pressure

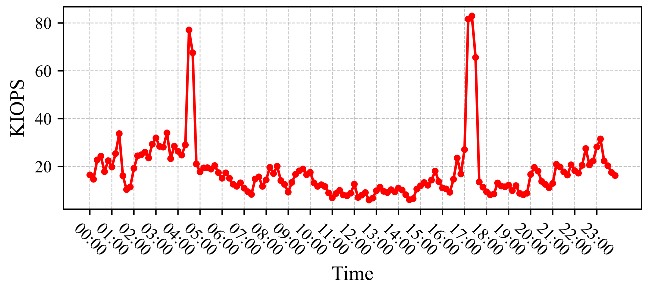

Fig. 3: IOPS of a storage server over 24 hours

Set Computing (CISC). RISC architectures feature a concise set of simple instructions that can be executed in a single clock cycle. This characteristic facilitates swift decoding and execution [37]. In contrast, CISC architectures utilize complex instructions capable of performing multiple operations. Consequently, they require fewer instructions to execute complex tasks, which can result in more memory-efficient code [38]. Traditionally, ARM processors excel in power efficiency and compact designs, while x86 processors offer higher computing power and wider software compatibility [39].

Prior works conducted extensive research on the power efficiency of x86 and ARM CPUs, and the feasibility of replacing x86 CPUs with ARM CPUs in data-centers [28], [40]–[42]. Early generations of ARM CPUs suffer from low performance, leading to extended task execution times and consequently higher power consumption compared to their x86 counterparts [28].

Modern, server-class ARM CPUs demonstrate significant improvements in performance and power efficiency compared to prior ARM CPU generations [43], [44]. Further, modern ARM CPUs offer superior power efficiency compared to x86 CPUs, delivering higher performance per unit area and watt [37]. While the single-core performance of ARM may be comparatively weaker than that of x86 [45], this limitation is typically mitigated by the architectural advantages of ARM of accommodating a higher number of cores within a single CPU die [27], [28], [46].

III. Analysis of ARM-based Storage Servers

Prior works analyze the power consumption of compute and HDD-based storage servers [21], [47], [48], compute servers with a single storage device [47], and investigate known power-saving techniques [49]. Additionally, other works note that different IO patterns lead to varying energy-efficiency in ARM SoC based *compute-servers* [42].

This section presents a comprehensive analysis of *all-flash storage servers* equipped with *server-class ARM CPUs* deployed across several data-centers. We present an analysis of the utilization patterns and power consumption in data-center storage servers. Additionally, we evaluate the impact of established power-saving techniques on IO performance and power draw of ARM-based storage servers.

A. Utilization and power consumption analysis

We analyze the utilization and power consumption of on-premise storage servers. Our analysis is based on data collected from more than 80 data-centers and institutions with storage servers that do not incorporate power-saving methods. These data centers involve those which serve banks, hospitals, financial centers, etc. The detailed hardware specifications of the analyzed systems are listed in §V.

1) Power consumption by component: We measure the power consumption of each hardware component in the storage server under a typical (low) load based on our analysis of storage server utilization in on-premise data-centers (workload is detailed in §III-A3). We also measure the power draw under high IO workload pressures. Fig. 2 shows the results.

We observe that without power-saving methods, the power draw of most components do not vary by a significant amount. The most power consuming components under the typical (low) loads are the CPUs and SSDs.

Observation 1: CPUs and SSDs collectively account for 49-60% of the total power consumption in storage servers, and power-saving strategies should prioritize these components to enhance system-wide power efficiency.

2) Utilization patterns of storage servers: The utilization of on-premise storage server varies throughout the day [50]–[52]. Fig. 3 shows the IOPS profile of a representative on-premise storage server from our sampled servers, captured over a 24-hour period. As depicted in the figure, the IOPS remain at a relatively low level throughout the day with occasional peaks.

Even in the absence of user-initiated IO requests, the system is never kept idle for long periods of time due to continuous background tasks which keep the system active. Based on the profiled data, we observe that on-premise storage servers perform numerous background tasks (§II), such as file-system garbage-collection, which can run continuously. These low-priority background tasks generate IO requests for the SSDs, which lead to continuous system activity throughout the day. Besides, several background tasks, such as flushing and compaction, can cause significant I/O and CPU bursts, leading to severe latency spikes [32]. Hence, it is necessary to restrict the resources allocated to background tasks, such as IO bandwidth and CPUs, to ensure adequate front-end performance for an optimal user experience [53].

It is worth noting that various dedicated roles for storage servers, such as backup storage servers, are only directly utilized for a fraction of the day to serve user IO requests [54]. These servers remain idle a prolonged period of time, but continuously perform necessary system background tasks.

Observation 2: In the absence of user-initiated IO requests, background tasks can continuously dispatch IO requests, keeping the system engaged.

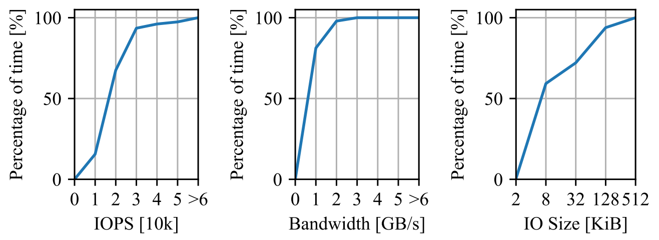

Fig. 4: cdf of IOPS, bandwidth, and IO request size over time

3) IO workload distribution in storage servers: We show the IOPS, IO size, and bandwidth distributions that a storage server handles over a period of 24-hours based on the same dataset as in §III-A2 in Fig. 4. These metrics, however, cannot individually represent the overall system load due to the complexity of storage servers and devices. Observing a single metric in isolation, such as IOPS or bandwidth, cannot represent the overall system load. These metrics are dependent on other IO characteristics, such as IO size, which in turn exert pressure on different components of the system, thereby affecting overall performance.

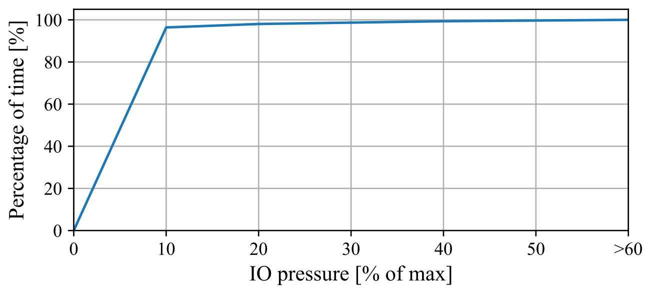

Fig. 5: cdf of IO pressure over time

Our goal in this analysis is to depict the overall system load, or "pressure" the storage system is under, based on the real-world traces as a function of time. This metric should represent the overall system load relative to its maximum performance across all of its resources.

To determine the overall system pressure based on real-world traces, we create a representative micro-benchmark that mimics the real-world traces in a controllable and configurable manner. First, we quantize the trace into 20 minute segments, and for each segment (t) we calculate: the average IOPS (I_t), read to write ratio (R_t), access pattern – random to sequential ratio (A_t), and IO size distribution (S_t). We then configure a micro-benchmark utility (see §V for more details) with the same parameters per 20 minute segment, and run for all segments twice: first we configure the IOPS to the average measured IOPS (I_t), then we run it without limiting the IOPS in order to achieve the maximum performance with the same IO characteristics. We calculate the system pressure as:

$$pressure = 100 \cdot \frac{\sum_{t=0}^{T} \sigma(I_t, R_t, A_t, S_t)}{\sum_{t=0}^{T} \sigma(max, R_t, A_t, S_t)}$$

Where σ is the micro-benchmark execution runtime.

Fig. 5 shows the cdf of the system pressure over time. The measured system pressure is under 10% for 95% of the time, depicting that it can potentially achieve 10× more IOPS with the same IO characteristics.

Observation 3: Storage servers spend the majority of the time under low IO pressure. Power-saving strategies must target this scenario to significantly lower the power consumption of storage servers.

B. Impact of power-saving methods on IO performance

1) CPU power-saving: Modern processors incorporate a Power Management Unit (PMU) that can adjust the power consumption of cores [55]. The PMU can be leveraged to enhance the cores' power efficiency by regulating energy consumption to accommodate the dynamic energy requirements of varying workloads.

We evaluate the performance and power consumption impact of two well-known power control modules provided by the PMU: Dynamic Voltage and Frequency Scaling (DVFS), and Power Gating (PG) [56] (see §V for evaluation methodology). We run a synthetic IO workload benchmark using vdbench with 8 KiB IO size. We perform random reads and writes at a ratio of 70:30, under 10% and 100% IO pressures, corresponding to 60k and 600k IOPS for this micro-benchmark, respectively. We measure the storage server's average and tail (99p) latencies, sustained IOPS, and IOPS/Watt:

Dynamic Voltage and Frequency Scaling. DVFS controls the power supply of all cores by adjusting the supplied voltage and frequency. Discrete voltage and frequency values are

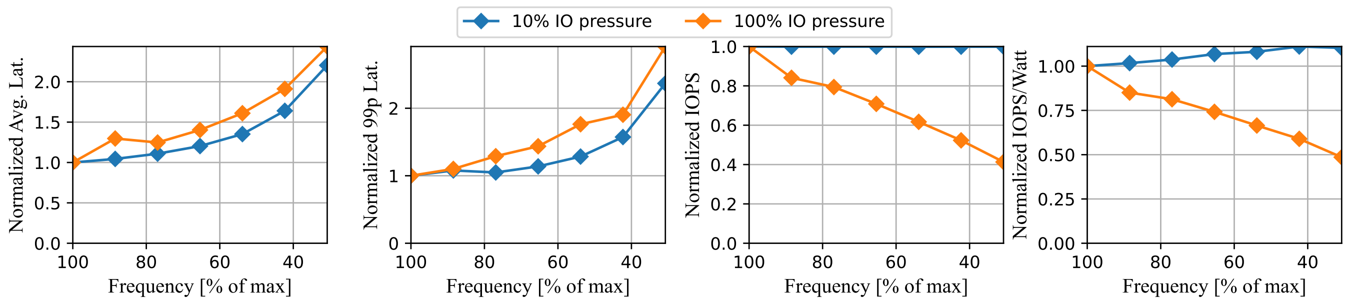

Fig. 6: Normalized average and 99p latencies, IOPS, and IOPS/Watt under different CPU frequencies

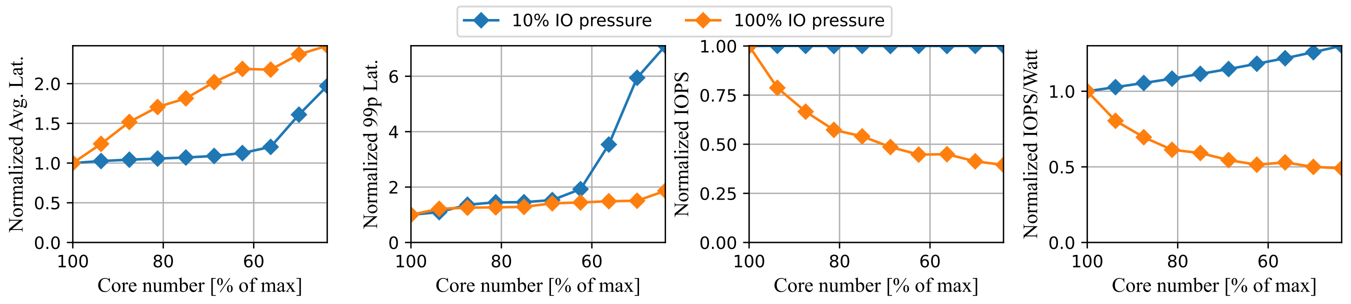

Fig. 7: Normalized average and 99p latencies, IOPS, and IOPS/Watt under different number of CPU cores

coupled together, thus they cannot be controlled individually. Higher frequency and voltage values usually result in higher performance, however they lead to higher energy consumption.

Fig. 6 shows the impact of DVFS on performance and power consumption by reporting the average and tail latencies, IOPS, and IOPS/Watt normalized to their corresponding values under 100% of the maximum CPU frequency.

We observe that DVFS affect both the average and tail latencies of the storage server under low and high IO pressures. This impact is noticeable even with a slight reduction in the CPU frequency, as the required operations the CPU needs to perform to serve IO requests take longer to complete.

Under low IO pressure, there is an increase in IOPS/Watt, signifying power savings. As the CPU is not fully utilized and is not the storage system's bottleneck, its frequency can be reduced to conserve power without impacting the IOPS. However, this reduction has implications for latency, as elaborated earlier.

In scenarios of high IO pressure, there is a reduction in IOPS/Watt. As the CPU operates at full capacity, a decrease in frequency results in the CPU becoming the storage system's bottleneck. Specifically, when the system is configured to run with a frequency of 30% of the maximum CPU frequency, DVFS leads to an average IO latency increase of up to 2.5×, a decline in sustained IOPS, and reduces the IOPS/Watt to 0.5× the IOPS/Watt under maximum frequency, thereby harming power efficiency.

Power gating. PG controls the power consumption of the processor by turning on and off individual CPU cores. When PG turns off a core, the core's voltage supply is cut off, thus it does not consume any energy. Tasks running on cores that are turned off are migrated to different active cores.

Fig. 7 shows the impact of PG on performance and power consumption by reporting the average and tail latencies, IOPS, and IOPS/Watt normalized to their respective values when all cores are active.

At low IO pressure, the reduction in the number of active cores does not immediately impact the average and tail latencies, in contrast to DVFS, where decreasing the CPU frequency leads to an immediate increase in IO latencies. This is because not all cores are fully utilized under low IO pressures. When the proportion of active cores falls below 63%, there is an increase in both average and tail latencies. The increase in latencies stems from the coarse-grained approach of deactivating entire cores for power-saving purposes. Turning off cores reduces the system's capacity to handle IO requests, akin to the impact observed when increasing throughput in a traditional IO system's throughput-latency experiment. The impact of PG on IOPS/Watt is more profound than DVFS, due to its ability to conserve more power by deactivating entire cores. We note that under 63% active cores, the IOPS/Watt increases by 1.18× with minimal IO latency impact.

Under 100% IO pressure, the IO tail latencies during the deactivation of cores closely resemble those observed with 100% active cores. This is because the tail latencies under heavy IO load increase in both scenarios. Additionally, the rise in the number of deactivated cores correlates with a decrease in sustained IOPS, resulting in a decrease in IOPS/Watt, similar to the effects observed in DVFS. The low tail latency under 100% IO load stands in contrast to the DVFS case, where the scaling down of all core frequencies leads to a longer completion time for each IO transaction, irrespective of the

SSD Load	Power	SSD power relative to 0%	Wake-up latency	Full-system power-saving
100%	13.3 W	1.56×	0 us	NA
50%	11.7 W	1.38×	0 us	NA
10%	10.5 W	1.24×	0 us	NA
0%	8.5 W	1×	0 us	0%
Idle-0 [1 us]	6.8 W	0.8×	1 us	6.7%
Idle-1 [200 us]	5.9 W	0.69×	200 us	10.3%

TABLE I: SSD power consumption under different IO pressures and power states, and their impact on full-system power consumption

IO load. This results in an immediate increase in both average and tail latencies.

Observation 4: Power-saving methods can harm both system performance and power efficiency if applied in unsuitable scenarios, such as during high IO pressures. PG exhibits minimal performance impact when deactivating unused cores, but is coarse-grained. In contrast, DVFS consistently affects IO latency but allows for fine-grained adjustments.

2) SSD power saving: Modern SSDs incorporate power-saving mechanisms, such as Autonomous Power State Transitions (APST) [57]. APST enables SSDs to transition between various power-saving states based on their operational workload. SSDs transition into a power-saving state following a specified duration of inactivity and adopt distinct power-saving states corresponding to varying durations of inactivity.

We quantify the power consumption of individual SSDs under varying IO pressures and power states, assessing their respective contributions to overall system power savings. Detailed results are presented in Table I.

The Idle-0 and Idle-1 states are activated following idle periods of 1 us and 200 us, respectively. Idle-0 can yield a total system power reduction of approximately 6.7% for the storage server, while Idle-1 achieves a 10.3% reduction, both accompanied by relatively low wake-up latencies.

Observation 5: SSDs enter power-saving states during idle periods, with minimal end-to-end latency impact during transitions. To enhance power efficiency in storage servers, we should maximize the SSDs' idle time.

C. Discussion

Traditional power-saving methods can be directly applied to ARM-based storage servers. However, the use of these methods can negatively impact system performance beyond a defined service-level agreement (SLA), which can vary between data-centers. Moreover, the efficacy of the power-saving methods is restricted if they are not designed in conjunction with the target application in a holistic manner, in this case a storage server.

Operating systems can utilize the Advanced Configuration and Power Interface (ACPI) to manage the power and performance of CPUs. ACPI offers control over the CPU's C-states – responsible for managing idle states, and P-states – overseeing core voltage and frequencies. In Linux, two power management subsystems can be utilized to conserve system power: `cpufreq` for P-states, and `cpuidle` for C-states.

`cpufreq` provides the on-demand governor [26] that can be used to save CPU power. It accomplishes this by setting the CPUs' target frequency using the calculation:

$$f_{target} = f_{max} \cdot T_{idle}/T_{total}$$

where f_{target} is the target frequency, T_{idle} is the time of being idle, and T_{total} is the sampling period. The absence of any storage-related metrics in its policy calculations (such as IO latency) renders it as an unsuitable candidate for energy-saving in storage servers, both in terms of adhering to a predefined SLA, and its power-saving abilities. We demonstrate its inefficacy for storage servers in §V.

The `cpuidle` mechanism activates various CPU idle states during periods when the operating system does not have scheduled tasks for the processor cores. However, the mechanism does not influence the operating system's scheduling algorithm to try to minimize the number of active cores in the system. `cpuidle` lacks a mechanism to consider the trade-off between performance and power in order to implement power-saving strategies. Additionally, it does not implement PG. In comparison to C-states, PG has demonstrated the potential to achieve a more substantial reduction in power consumption, nearing 50% as demonstrated in prior work [58].

Another example of the constraints of generic power-saving methods in storage servers is the use of SSD's APST. APST achieves maximum power savings when there are complete idle periods. As shown in §III-A2, storage servers generally run background tasks throughout the day (§III-A2), limiting the power-saving potential of APST.

Based on the observations in this section, we conclude that there are ample opportunities to apply power-saving methods to the most power-consuming components in on-premise storage servers (observations 1-3). Furthermore, power-saving methods need to be utilized in conjunction with the storage server scenario to maintain predefined SLAs while maximizing power-saving opportunities (observations 4-5).

IV. Design and Implementation

Our design goals for PACER are as follows:

Power-saving mechanism for storage servers. The power-saving techniques in PACER should be designed in conjunction with the target application – storage servers. Such a design that leverages the characteristics of storage servers and utilizes IO-specific metrics for power-saving, can attain a high level of power-saving while minimizing the impact on performance.

Adjustable SLA guarantee. Power-saving methods incur system performance losses, some of which cannot be mitigated. Storage server deployments in different data-centers may have different SLAs, performance, and power considerations. To accommodate such scenarios, PACER should expose the performance-to-power saving tradeoff to the system administrator in an intuitive and useful manner. Storage system administrators should be able to specify maximum tolerable performance loss that is still within the SLA envelope based on IO performance metrics.

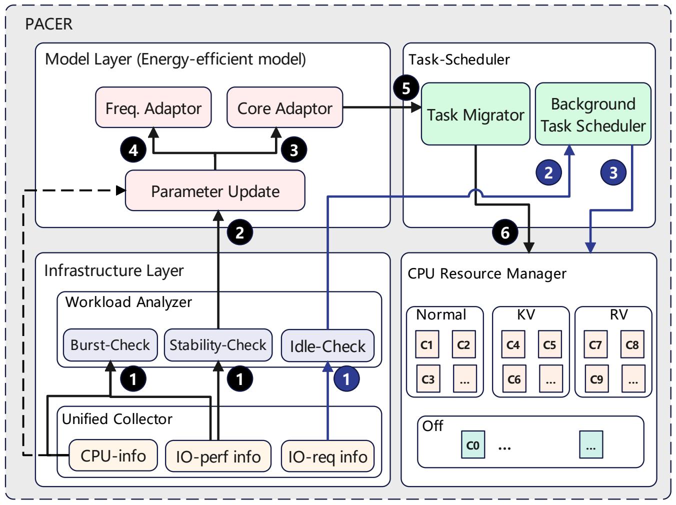

Fig. 8: Overview of PACER

ARM-friendly software design. ARM-based server-grade CPUs exhibit superior performance-to-power ratios over their x86 counterparts [38], hence PACER builds upon the ARM architecture for higher power efficiency. However, a single ARM core is usually computationally weaker than an x86 core [28]. The disparity in single-core performance is usually counteracted by the number of cores a single ARM-based CPU can contain [46]. PACER should be designed for this scenario.

A. Design overview

PACER's main design components are shown in Fig. 8. We explain each of the components and its role during PACER's operational state.

PACER applies power-saving methods during stable system IO load states exclusively. Applying these methods during unstable IO load periods can lead to undesirable fluctuations in IO performance. The *Stability-Check* and *Burst-Check* components within the *Workload Analyzer* actively monitor the system stability, utilizing the *CPU-info* and *IO-perf info* components of the *Unified Collector* (❶) to obtain CPU and IO system metrics.

During stable IO periods, the *Parameter Update* component within the *Model Layer* establishes a performance baseline under the current IO workload by gathering information from the *Infrastructure Layer* (❷). The *Core Adaptor* then calculates the minimum number of cores necessary to sustain the current IO performance (❸), while adhering to a user-configured SLA envelope. Then, the *Freq. Adaptor* (❹) fine-tunes the power-saving by adjusting the CPU core frequency.

The *Task Migrator* redistributes tasks from cores designated for deactivation to other active cores (❺). The *CPU Resource Manager*, responsible for the efficient distribution and scheduling of tasks across cores, deactivates the appropriate cores (❻).

This process is performed iteratively with feedback obtained from the *Unified Collector*.

The *Idle-Check* assesses the system's idle status by monitoring the incoming IO requests through *IO-req info* (❶). Upon recognizing the system as idle, it activates the *Background Task Scheduler* to optimize the scheduling of background tasks, aiming to achieve additional power savings during inactive periods (❷).

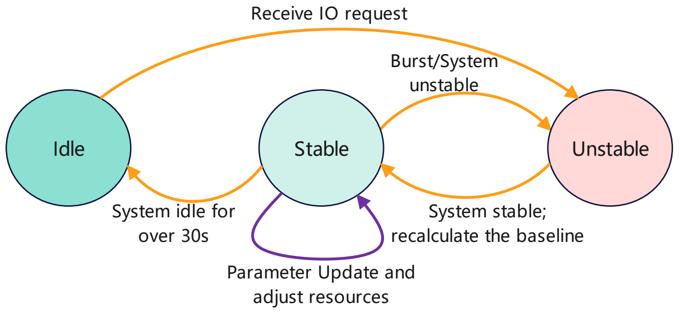

Fig. 9: PACER subsystem state machine

B. Storage server stability monitoring

PACER monitors the system and designates a state based on IO and CPU metrics. Fig. 9 shows the different states and state transitions. PACER starts in the Unstable state.

Unstable state. PACER does not apply power-saving methods when it is in the Unstable state.

The *Stability Check* component can transition the system into the Stable state. It relies on IO latency metrics to determine whether the system is stable. Unlike metrics such as IOPS, which may exhibit relative fluctuations during low workload pressure, the latency does not fluctuate unless the system bottleneck has been reached [31]. Furthermore, latency is typically more crucial in low workload pressure scenarios [31].

Stability Check monitors the IO latency over a sliding time window: it samples the average latency of IO requests within each n second window (l_i), and calculates the average latency over k windows, and the divergence:

$$l_{avg} = \sum_{i=0}^{k-1} \frac{l_i}{k}, \qquad l_{div} = \frac{\sqrt{\sum_{i=0}^{k-1} \left(l_i - l_{avg}\right)^2 / k}}{l_{avg}}$$

If the divergence is less than a predefined threshold: $l_{div} < l_{stable\_thresh}$, the system is considered stable and transitions to the Stable state.

Stable state. While in the Stable state, PACER applies power-saving methods, detailed in §IV-C.

The *Stability-Check* component continuously monitors the latency divergence of the system. If the divergence exceeds the unstable threshold: $l_{div} < l_{unstable\_thresh}$, the system transitions to the Unstable state. Note that $l_{stable\_thresh} < l_{unstable\_thresh}$ is enforced to achieve hysteresis, so the PACER does not frequently transition between the states.

The *Burst-Check* component monitors potential IO bursts that may occur. These bursts may happen suddenly in a short period of time; shorter than the duration of the moving time window of the *Stability-Check* component. These bursts require rapid response to prevent adverse effects on system performance. The mechanism samples the CPU utilization and IO latency in short 500ms windows. If the measured latency divergence is larger than l_{burst_thresh} or the CPU utilization surpasses 80%, a threshold determined empirically to indicate system bursts, the component transitions the system to the Unstable state, deactivating power-saving measures.

PACER can transition into the Idle state when the *Idle-Check* component detects a period of at least 30 seconds without any incoming user IO requests. We have chosen a period of 30 seconds based on empirical evidence demonstrating stable behavior.

Idle state. During the Idle state, PACER initiates the *Background Task Scheduler* detailed in §IV-D.

The system transitions from Idle into the Unstable state upon receiving an incoming IO request from clients. PACER transitions to the Unstable state as it needs to assess the new operational state of the system and adapt to the new IO workload before transitioning back to the Stable state.

C. Performance monitoring and power-saving methods

In the Stable state, the IO latency is continuously monitored, averaged, and its divergence is calculated using a sliding window as stated in §IV-B. As long as $l_{div} < l_{stable\_thresh}$, the *Parameter Update* component in the *Model Layer* utilizes the current average IO latency (l_{avg}) as the baseline latency ($l_{baseline}$). The baseline is used to estimate the current performance loss resulting from the application of power-saving methods. This operation happens iteratively when PACER is in the Stable state and continuously tunes the power-saving methods. The performance loss is computed based on the latency measured in the current window, l_{cur}, and the baseline latency $l_{baseline}$:

$$loss_{cur} = \frac{l_{cur} - l_{baseline}}{l_{baseline}}$$

The calculated relative distance between the current performance loss ($loss_{cur}$) and the user-defined *performance-loss factor* (`plf`) denotes the acceptable threshold for performance compromise that the user is willing to tolerate for power-saving opportunities.

$$loss_{distance} = \frac{plf - loss_{cur}}{plf}$$

We employ $loss_{distance}$ to implement a combination coarse-grained (PG) and fine-grained (DVFS) power-saving methods. These methods are applied iteratively as long as the system remains in the Stable state. Note that $loss_{distance}$ is tied to IO performance degradation, which directly guides the CPU power-saving mechanisms.

Coarse-grained power saving. The *Core Adaptor* component within the *Model Layer* calculates the number of cores the storage system needs to employ while staying within the user-defined performance-loss factor.

Reducing the available number of cores will initiate the migration of system tasks from the affected cores to others, leading to an overall increase in CPU utilization attributable to the decreased number of active cores. *Core Adaptor* calculates the desired overall CPU utilization; a higher desired utilization value correlates to less overall active cores:

$$u_{desired} = u_{desired,old} + \beta \cdot loss_{distance}$$

The parameters $u_{desired}$ and $u_{desired,old}$ represent the desired new and old utilization, respectively. The configurable hyper-parameter β governs the aggressiveness of the *Core Adaptor*.

We observe the following phenomenon while determining the number of cores required to achieve the desired CPU utilization: as cores are deactivated in the system, the total utilization of the remaining cores following task migration exhibits a climb with a small slope. That is,

$$\sum_{i=0}^{n'} u_i \approx \alpha \cdot \sum_{i=0}^{n} u_i$$

Where n and n' are the number of cores before and after adjustment, respectively, and u_i is the utilization of core i. α is the measured slope.

Based on this observation, we calculate the number of desired cores ($n_{desired}$) using the current number of cores (n_{cur}), the current CPU utilization (u_{cur}), and the desired CPU utilization ($u_{desired}$):

$$n_{desired} = \lceil \alpha \cdot n_{cur} \cdot u_{cur} / u_{desired} \rceil$$

After configuring the system with the desired number of cores, the total utilization of the active cores will converge towards $u_{desired}$.

Fine-grained power saving. After establishing the number of cores in the system, the *Freq. Adaptor* refines the system's power consumption, aligning it with the user-defined maximum performance loss factor (`plf`) by finely-tuning the frequencies of the active cores.

PACER iteratively adjusts the core frequency based on the IO-driven $loss_{distance}$:

$$f_{desired} = f_{cur} + \gamma \cdot loss_{distance}$$

Where γ is a configurable hyperparameter that controls the aggressiveness of the *Freq. Adaptor*.

We demonstrate the individual and combined effects of coarse- and fine-grained power-saving mechanisms on the performance and power consumption of the storage system in Section V-A3.

D. Managing tasks

The *Task Migrator* migrates tasks from cores slated for deactivation to other active cores. It prioritizes the completion of ongoing tasks on the designated cores and reroutes new tasks to alternative cores within the same core group.

The *Background Task Scheduler* prioritizes background tasks when the system transitions into the Idle state. Specifically, it prioritizes ongoing background tasks, such as garbage-collection, by executing them at maximum performance, expediting their completion. This enables the system to enter an idle period with no ongoing or pending background tasks, facilitating the SSDs to transition into their idle-states and thereby optimizing overall system power-savings.

E. Managing CPU resources

PACER utilizes an event-based, run-to-completion model for managing network and IO requests within the system.

Server-grade ARM CPUs feature a higher number of less powerful cores compared to x86-based CPUs in order to achieve equivalent overall performance [27], [28], [45], [46]. To manage the numerous cores in server-grade ARM-based CPUs, the *CPU Resource Manager* organizes the available cores into groups, each assigned a specific group of tasks. This method bounds the number of cores contending for the same locks and shared data structures, reduces false cache-line sharing, which leads to higher performance [59]. The cores are divided into the following groups:

Normal core group handles incoming IO requests, carries out the initial IO processing (data service layer in Fig. 1), and performs data lookup in the cache (global cache layer).
KV or "Key-Value" cores are responsible for translating the address of IO requests from logical to physical (space management layer).
RV or "Resource Volume" cores are dedicated to managing physical disk resources and executing the IO operations on the physical device (space management and storage pool layers).

The number of cores within each group is dynamically adjusted utilizing the methods outlined in §IV-C. This adaptive approach is necessary because workloads may unevenly utilize distinct core groups; for instance, a workload with a high cache hit rate may necessitate scaling up the "Normal" cores.

V. Evaluation

Our evaluation demonstrates the power-saving capabilities of PACER while maintaining high system performance. We also demonstrate PACER's ability to navigate the power-performance tradeoff, and compare its power and performance metrics against other system configurations under different IO access patterns and end-to-end applications.

Methodology. we assess the performance and power employing following methodologies:

Performance. PACER leverages P-state assignment for frequency adjustment of all active cores, and CPU hotplug [60] for PG as utilized in prior works [61]. These mechanisms directly affect the measured performance metrics, and we evaluate our system performance utilizing these mechanisms.

Power. In our evaluation, we employ a server-grade ARM CPU that does not support power gating. However, upcoming iterations of the utilized CPU incorporate this technology. The precision of the performance evaluation stems from the direct impact of the mechanisms on system performance and the operations required to deactivate cores and control the frequency, supported by the current CPU generation.

In order to calculate the overall system power consumption, we measure the entire system power consumption using a power-meter and assess the per-component power consumption using the on-board BMC component. To accurately emulate the power draw of the CPU under PG and DVFS, we model the power draw using established power modelling techniques which has been utilized in prior works [62]:

$$P = C \cdot f \cdot V^2 + P_{static},$$

where V is the voltage, P_{static} is the static core power consumption, f is the frequency, and C is the CPU-specific power-coefficient.

First, we calculate the power-savings resulting from DVFS. We measure the power consumption of the CPU using the on-board Baseboard Management Controller (BMC) across various frequency values that PACER can set for the CPU. We find that at the measured frequencies, the voltage remains constant. Therefore, through linear regression, we extract the value of $\delta = CV^2$ and subsequently calculate the power-savings attributed to DVFS:

$$\begin{aligned} P_{cpu,freqsaving} &= \delta f_{max} + P_{static} - (\delta f + P_{static}) \\ &= \delta(f_{max} - f) \end{aligned}$$

We then determine the total power consumption of the CPU across different configurations of active cores. This involves conducting power measurements using the on-board BMC, which provides insights into the static and dynamic power consumption of specific CPU components, such as the core and other components. The overall CPU power consumption is then measured for varying numbers of active cores, configured through the BIOS settings. Notably, the dynamic components of the CPU power consumption exhibit linear scaling with the number of cores, aligning with established models found in prior works [63]:

$$P_{cpu} = P_{core} \cdot (1 - n/n_{max}) + P_{cpu,rest}$$

Where n and n_{max} are the number of active and overall cores, respectively. P_{core} and $P_{cpu,rest}$ are extracted using linear regression. All employed linear regressions exhibit a coefficient of determination exceeding 0.99.

The overall system power consumption is calculated:

$$P = (P_{cpu} - P_{cpu,freqsaving}) + P_{rest}$$

Execution. Each evaluated workload runs for 5 minutes, and we report the average and tail latencies, as well as IOPS/Watt, normalized to native execution. For the vdbench workload, three client machines are employed, each running 300 threads, effectively saturating the storage server. The IO pressure for both synthetic workloads and real-world traces is determined and replicated using the methodology described in §III-A3.

Testbed. We use a commercially available off-the-shelf storage server (Huawei OceanStor Dorado 6000) equipped with 2×ARMv8-based CPUs. Each CPU has 48 cores with a maximum frequency of 2.6 GHz. The total per-CPU cache sizes are: 64 MiB LLC, 24 MiB L2, 3 MiB L1d and 3 MiB L1i. Each socket has 32 GiB of DDR4 memory. The server houses 24 SSDs in a RAID-6 configuration, and the server runs a commercial storage system software stack. The 3 client machines that drive the storage server for the evaluation are connected via a 64 GiB/s FC link each.

Workload	Description
Rand_R	Uniform random 8 KiB reads
Rand_W	Uniform random 8 KiB writes
Seq_R	Sequential 128 KiB reads
Seq_W	Sequential 128 KiB writes
Rand_RW	Mix of 70:30 8 KiB reads and writes

TABLE II: vdbench IO access pattern configurations

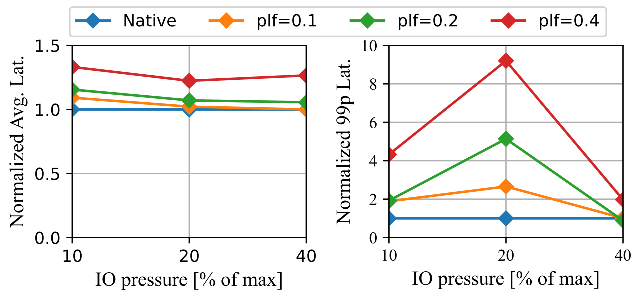

Fig. 10: Relative average and 99p IO latency over Native for different `plf` values under varying IO pressures. vdbench with Rand_RW access pattern (lower is better)

Configurations. We evaluate the storage server's performance and power consumption under the following configurations:
(a) Native: No power-saving strategies. Serves as our ideal upper bound on performance.
(b) Ondemand: Linux's "on-demand" power-saving governor.
(c) PACER: With `plf` of 0.2 (unless stated otherwise).

A. Synthetic Benchmarks

We evaluate PACER and the other system configurations on synthetic workloads using vdbench. The workload configurations are listed in Table II.

1) Effect of `plf` on power and performance: We evaluate the effects the hyperparameter `plf` that controls the performance loss tolerance factor has on system performance and power. We run vdbench using the Rand_RW configuration with PACER under different IO workload pressures. We then vary the value of `plf`, and measure the IO latency and power consumption. We also evaluate the performance of Native under different IO pressures, using it as our baseline and ideal upper-limit on performance. We measure the results of PACER relative to Native.

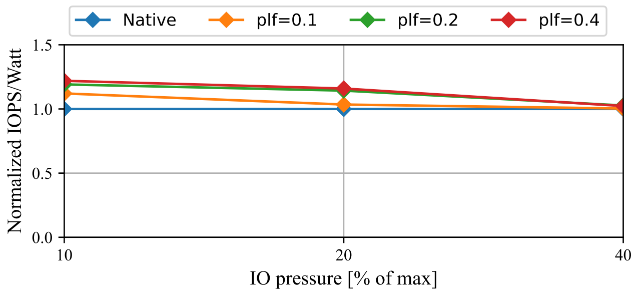

Fig. 11: Relative IOPS/Watt over Native of the entire storage server for different `plf` values under varying IO load pressures. vdbench with Rand_RW access pattern (higher is better)

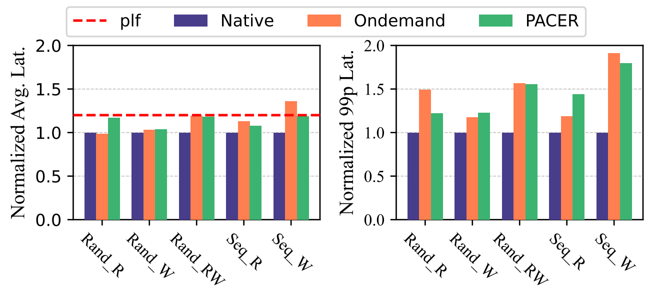

Fig. 12: Relative average and 99p IO latency over Native for different configurations and access patterns (lower is better)

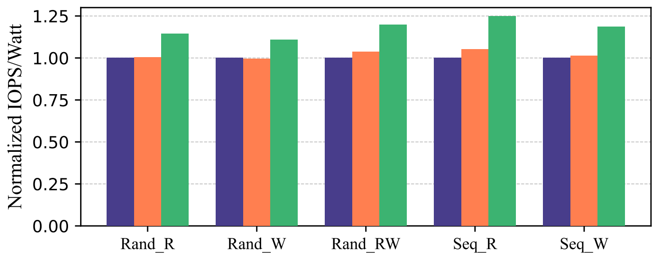

Fig. 13: Relative IOPS/Watt over Native of the entire storage server for different configurations and access patterns (higher is better)

Fig. 10 and 11 show the impact of `plf` on IO latency and IOPS/Watt of the entire storage server over Native, respectively. We observe that the measured average latencies are less than the configured `plf` for each pre-configured `plf` value. As the IO load intensifies, the relative IOPS/Watt decreases, showcasing PACER's adaptive adjustment of power-saving methods to the IO load, subsequently throttling them down. The adaptation reduces the impact of power-saving methods on performance, ensuring compliance with the configured performance loss tolerance. We note that under the highest evaluated IO load of 40%, the 99p latency of PACER relative to Native decreases. This trend is attributed to the increase in 99p latency of the Native system at this level, and PACER refraining from aggressive power-saving measures. The sustained IOPS is similar for all `plf` values and Native (not shown).

We assess the tradeoff between the actual performance loss and the IOPS/Watt benefits across various `plf` values to identify the most advantageous setting. Our empirical analysis indicated that a `plf` value of 0.2 strikes a favorable balance between performance and power considerations. Specifically, a `plf` value of 0.2 exhibited a maximum latency increase of 15% with 20% power saving for typical IO pressures (§III-A3).

2) Impact of IO access patterns: Previous research [41], [42] highlight the impact of different IO access patterns on performance and power on compute servers. We evaluate this impact on our storage server with PACER, under the different IO patterns listed in Table II, and measure the average and tail IO latencies, and IOPS/Watt.

Fig. 12 and 13 show the relative increase in IO latency

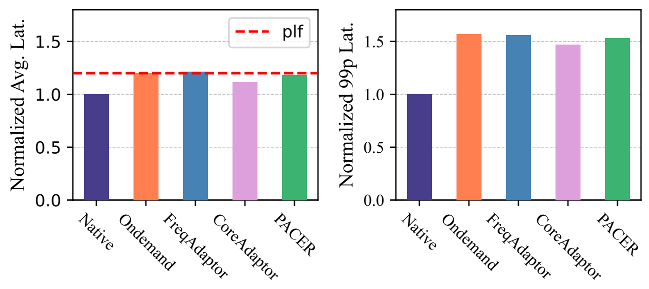

Fig. 14: Relative average and 99p IO latency over Native for different configurations and power-saving methods. vdbench with Rand_RW access pattern (lower is better)

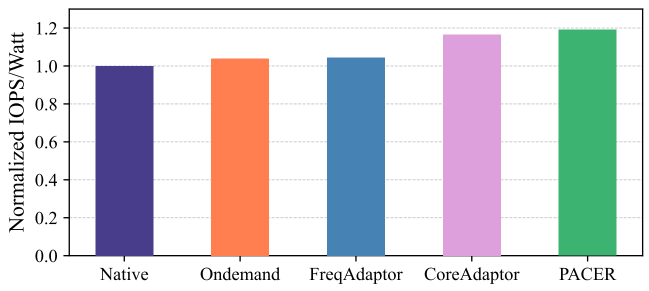

Fig. 15: Relative IOPS/Watt over Native of the entire storage server for different configurations and power-saving methods. vdbench with Rand_RW access pattern (higher is better)

and IOPS/Watt of the entire storage server over Native, respectively, for PACER and Ondemand. The sustained IOPS is similar for all configurations (not shown). Notably, different IO access patterns exert varying effects on the achieved power-saving and IO latency increases of PACER. First, we note that the average latency increase of PACER remains within the predetermined `plf` for all access patterns. While PACER and Ondemand exhibit comparable impacts on tail latencies, PACER conserves more power, with a geometric mean increase in IOPS/Watt of 1.18× across the access patterns, surpassing Ondemand's 1.02× increase.

3) Analysis of individual power-saving components: We evaluate the individual impact of each power-saving component in PACER on both power and performance by enabling them one-by-one. We compare the methods to Native and Ondemand.

Fig. 14 and 15 show the impact of the different methods on IO latency, and IOPS/Watt of the entire storage server, respectively. IOPS is similar for all methods (not shown). Ondemand achieves a modest 1.02× increase in IOPS/Watt, and exhibits a greater average and tail IO latencies when compared to PACER. Ondemand has similar behavior to the FreqAdaptor component, as they both rely solely on DVFS for operation. Additionally, we observe that the CoreAdaptor component (PG) has a more profound impact on power-saving than FreqAdaptor (DVFS), as corroborated in §III-B1. While CoreAdaptor operates at a coarse-grained level, PACER complements it with the fine-grained FreqAdaptor to achieve a

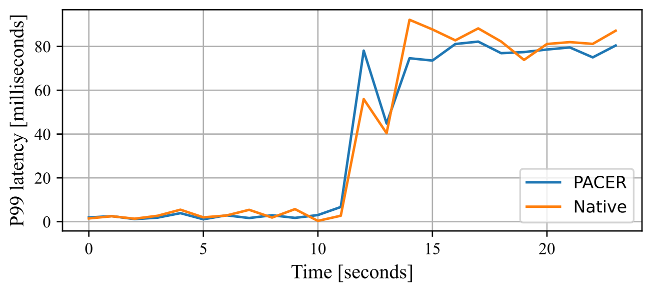

Fig. 16: 99p latency over time during a burst in IO requests

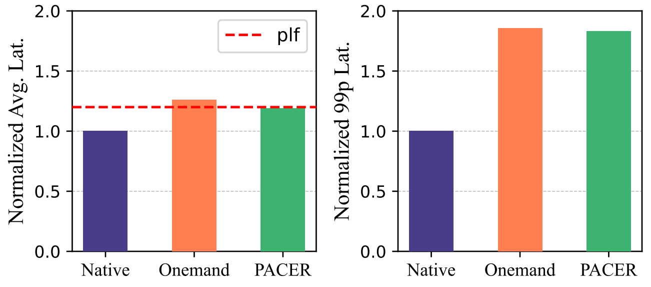

Fig. 17: Relative average and 99p IO latency over Native for real-world trace (lower is better)

higher IOPS/Watt compared to each individual approach. This combination results in lower IO latency than the maximum observed when using either method in isolation.

4) Analysis of IO bursts: IO bursts can impose stress on the system through a sudden influx of IO operations, and the implementation of power-saving measures may potentially impede performance and responsiveness during such bursts.

To assess PACER's response to IO bursts and its Burst-check component, we execute `vdbench` using the Rand_RW IO access pattern. Initially, it is executed with 3% IO pressure to allow PACER to stabilize and transition to the Steady state. Following a 10-second waiting period, the IO size is then increased to 32 KiB, and the IO pressure is raised to 100%. The 99p tail latency for IO requests is then measured at one-second intervals. Fig. 16 shows the results.

During the low IO pressure period, PACER employs power-saving measures, including PG and DVFS. The Workload Analyzer components (Fig. 8) continuously monitor the system state and incoming IO requests. Upon transitioning to a 100% IO pressure rate, the fine-grained Burst-Check component detects the surge in IO requests and promptly shifts the system into the Unstable state. This entails activating all CPU cores, and setting their frequencies to the maximum value. The sustained IOPS for PACER throughout the experiment is similar to that of Native (not shown)

Due to the prompt response of the Burst-Check component, both PACER and Native exhibit comparable increases in 99p latencies during the burst, reaching similar steady-state values.

B. Real-world benchmarks

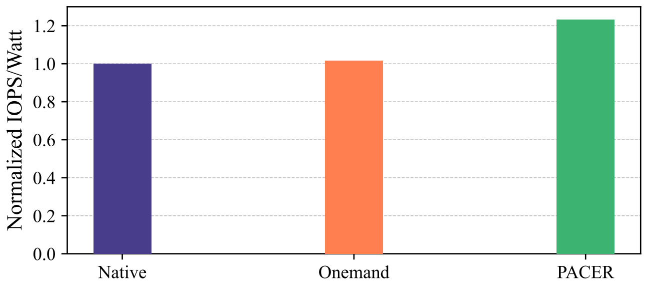

Fig. 18: Relative IOPS/Watt over Native of the entire storage server for real-world trace (higher is better)

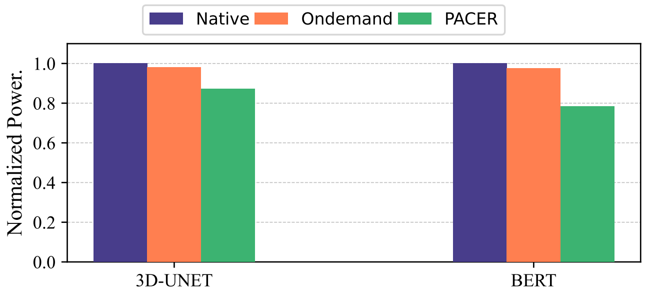

Fig. 19: Relative power consumption over Native of the entire storage server for MLPerf Storage benchmark suite (lower is better)

1) Real-world traces: We evaluate PACER using a real-world trace collected from multiple data-centers, as detailed in §III. We compare PACER to the Ondemand and Native configurations. Fig. 17 depicts the effect on the average and 99p IO latency relative to Native, and Fig. 18 illustrates the impact on IOPS/Watt of the entire storage server relative to Native. The IOPS is similar for all configurations (not shown).

PACER achieves power savings while conforming to its pre-defined performance loss tolerance of 20%. It demonstrates a 1.23× increase in IOPS/Watt of the entire storage server over Native, and a 1.21× higher IOPS/Watt compared to Ondemand. PACER experiences a 1.19× increase in average IO latency over Native, adhering to the pre-configured `plf` hyperparameter. Crucially, the rise in both average and tail IO latencies for PACER is lower than Ondemand's increase. Ondemand exhibits 1.26× higher average IO latency over Native, while only achieving a modest 1.02× higher IOPS/Watt, underscoring its limited suitability for storage servers.

2) MLPerf Storage: We evaluate PACER using MLPerf Storage benchmark suite [33] that characterizes the performance of storage systems for machine learning workloads. The benchmark suite consists of two separate workloads: image segmentation with 3D-UNET, and natural language processing with BERT. We run MLPerf Storage on a client machine configured with 16 accelerators and measure the performance and power consumption for each workload execution.

Both Ondemand and PACER are able to attain the same performance levels as Native, as the benchmark's performance metrics are throughput oriented. However, the power-savings of the configurations differ. As illustrated in Fig. 19, PACER achieves an overall system power consumption reduction of 13% and 22% for 3D-UNET and BERT, respectively. In contrast, Ondemand achieves more modest power savings of only 2% and 2.5% of power savings for 3D-UNET and BERT respectively.

VI. Related Work

ARM and storage. Prior works demonstrate that ARM-based architectures exhibit higher power efficiency than x86 [40], [41], [64], with recent works exploring the use ARM-based CPUs and their effect on IO performance [30], [44], [65]. Pavan et al. [42] asserts that due to the lower performance of ARM cores compared to x86, the IO throughput of storage devices utilizing ARM is lower than those based on x86. However, modern ARM CPUs that are natively designed for cloud servers can provide comparable IO performance to x86 when the software stack is optimized [40], [43], [44]. Li et al. [64] designed a heterogeneous system based on x86 and ARM for storage devices. The ARM cores in the system are integrated in NIC and complement the x86.

These works primarily focus on raw IO performance for compute systems. Our work complements these studies by examining and optimizing power management of ARM-based storage servers.

Traditional power-saving methods. The optimization of power efficiency in data-centers has been analyzed from multiple perspectives. The efficiency of individual components in data-center servers, including CPUs [55], [56], GPUs [23], DRAM [66], [67], and SSDs [15], [68], has been thoroughly studied to identify opportunities for improvement. Several works analyze the characteristics of application workloads and improve the energy efficiency using task scheduling techniques [58], [61], [69].

Our work characterises and models the power and performance of storage servers. PACER is a power-saving system that achieves high power efficiency by leveraging power-saving methods for storage servers.

Power saving in storage devices. Active flash [15] reduces power consumption by minimizing data movements between the SSD and CPU. It performs processing in storage by offloading data analysis to SSD controllers. Alternative approaches involve redirecting IO request to a limited subset of storage devices, enabling the other devices to benefit extended periods of inactivity, thereby conserving power [48], [70]–[73]. Satoshi et al. leverage DVFS to reduce the power consumption of compute systems that employ Ultra-Low Latency (ULL) SSDs [74]. Their major observation is that the CPU core frequency can be *reduced* while the ULL SSDs are *active*.

These works focus on improving the power efficiency of specific components or modules. PACER takes a systematic approach to power management in storage systems and considers numerous opportunities for power savings.

Power saving in databases. Prior works focus on energy modelling of database systems, and predict and profile the power consumption of database queries [75]–[78]. DVFS for database systems has been explored in prior works for enhanced power efficiency [79], [80]. Notably, these approaches are tailored for frequency selections within cloud database clusters, leveraging specific characteristics of queries and transactions. However, their applicability in the storage server scenario is limited due to their distinct operational requirements.

VII. Conclusions

PACER is a power-saving subsystem designed for ARM-based, all-flash storage servers. PACER's design is driven by a thorough analysis of storage server utilization patterns and power consumption across various data-centers. Unlike conventional approaches, PACER is guided by storage server-specific IO metrics, employing advanced monitoring to assess operational conditions and predict the potential impact of power-saving techniques on IO performance. This allows PACER to navigate the delicate balance between performance and power-efficiency by adhering to configurable boundaries.

References

[1] M. Dayarathna, Y. Wen, and R. Fan, "Data Center Energy Consumption Modeling: A Survey," *IEEE Communications surveys & tutorials*, vol. 18, no. 1, pp. 732–794, 2015.

[2] V. Sakalkar, V. Kontorinis, D. Landhuis, S. Li, D. De Ronde, T. Blooming, A. Ramesh, J. Kennedy, C. Malone, J. Clidaras, and P. Ranganathan, "Data Center Power Oversubscription With a Medium Voltage Power Plane and Priority-Aware Capping," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 497–511.

[3] N. Sukhija, E. Bautista, D. Butz, and C. Whitney, "Towards Anomaly Detection for Monitoring Power Consumption in HPC Facilities," in *Proceedings of the 14th International Conference on Management of Digital EcoSystems*, 2022, pp. 1–8.

[4] Y. Shi, N. Nie, J. Wang, K. Lin, C. Zhou, S. Li, K. Yao, S. Li, Y. Feng, Y. Zeng, F. Liu, Y. Wang, and Y. Gao, "Large-Scale Simulation of Structural Dynamics Computing on GPU Clusters," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2023.

[5] J. Sevilla, L. Heim, A. Ho, T. Besiroglu, M. Hobbhahn, and P. Villalobos, "Compute Trends Across Three Eras of Machine Learning," in *2022 International Joint Conference on Neural Networks (IJCNN)*, 2022, pp. 1–8.

[6] M. Zhao, N. Agarwal, A. Basant, B. Gedik, S. Pan, M. Ozdal, R. Komuravelli, J. Pan, T. Bao, H. Lu, S. Narayanan, J. Langman, K. Wilfong, H. Rastogi, C.-J. Wu, C. Kozyrakis, and P. Pol, "Understanding Data Storage and Ingestion for Large-Scale Deep Recommendation Model Training: Industrial Product," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, 2022, pp. 1042–1057.

[7] J. Do, V. C. Ferreira, H. Bobarshad, M. Torabzadehkashi, S. Rezaei, A. Heydarigorji, D. Souza, B. F. Goldstein, L. Santiago, M. S. Kim, P. M. V. Lima, F. M. G. França, and V. Alves, "Cost-Effective, Energy-Efficient, and Scalable Storage Computing for Large-Scale AI Applications," *ACM Trans. Storage*, vol. 16, no. 4, 2020.

[8] B. Nicolae, J. Li, J. M. Wozniak, G. Bosilca, M. Dorier, and F. Cappello, "DeepFreeze: Towards Scalable Asynchronous Checkpointing of Deep Learning Models," in *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, 2020, pp. 172–181.

[9] A. K. Paul, J. Y. Choi, A. M. Karimi, and F. Wang, "Machine Learning Assisted HPC Workload Trace Generation for Leadership Scale Storage Systems," in *Proceedings of the 31st International Symposium on High-Performance Parallel and Distributed Computing*, 2022, pp. 199–212.

[10] W. Chen, S. He, Y. Xu, X. Zhang, S. Yang, S. Hu, X.-H. Sun, and G. Chen, "iCache: An Importance-Sampling-Informed Cache for Accelerating I/O-Bound DNN Model Training," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2023, pp. 220–232.

[11] R. I. S. Khan, A. H. Yazdani, Y. Fu, A. K. Paul, B. Ji, X. Jian, Y. Cheng, and A. R. Butt, "SHADE: Enable Fundamental Cacheability for Distributed Deep Learning Training," in *21st USENIX Conference on File and Storage Technologies (FAST 23)*, 2023, pp. 135–152.

[12] D. Zha, Z. P. Bhat, K.-H. Lai, F. Yang, Z. Jiang, S. Zhong, and X. Hu, "Data-centric Artificial Intelligence: A Survey," 2023.

[13] A. K. Paul, A. M. Karimi, and F. Wang, "Characterizing Machine Learning I/O Workloads on Leadership Scale HPC Systems," in *2021 29th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2021, pp. 1–8.

[14] D. Saxena, J. Kumar, A. K. Singh, and S. Schmid, "Performance Analysis of Machine Learning Centered Workload Prediction Models for Cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 4, pp. 1313–1330, 2023.

[15] D. Tiwari, S. Boboila, S. S. Vazhkudai, Y. Kim, X. Ma, P. J. Desnoyers, and Y. Solihin, "Active Flash: Towards Energy-Efficient, In-Situ Data Analytics on Extreme-Scale Machines," in *11th USENIX Conference on File and Storage Technologies*, 2013, pp. 119–132.

[16] T. Jiang, G. Zhang, Z. Huang, X. Ma, J. Wei, Z. Li, and W. Zheng, "FusionRAID: Achieving Consistent Low Latency for Commodity SSD Arrays," in *19th USENIX Conference on File and Storage Technologies (FAST 21)*, 2021, pp. 355–370.

[17] J. Koo, J. Bae, M. Yuk, S. Oh, J. Kim, J.-S. Park, E. Lee, B. S. Kim, and S. Lee, "All-Flash Array Key-Value Cache for Large Objects," in *Proceedings of the Eighteenth European Conference on Computer Systems*, 2023, pp. 784–799.

[18] L. Shi, L. Luo, Y. Lv, S. Li, C. Li, and E. Hsing-Mean Sha, "Understanding and Optimizing Hybrid Ssd With High-Density and Low-Cost Flash Memory," in *2021 IEEE 39th International Conference on Computer Design (ICCD)*, 2021, pp. 236–243.

[19] E. Xu, M. Zheng, F. Qin, Y. Xu, and J. Wu, "Lessons and Actions: What We Learned From 10K SSD-Related Storage System Failures," in *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, 2019, pp. 961–976.

[20] C. Min, S.-W. Lee, and Y. I. Eom, "Design and Implementation of a Log-Structured File System for Flash-Based Solid State Drives," *IEEE Transactions on Computers*, vol. 63, no. 9, pp. 2215–2227, 2014.

[21] E. Tomes and N. Altiparmak, "A Comparative Study of HDD and SSD RAIDs' Impact on Server Energy Consumption," in *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, 2017, pp. 625–626.

[22] T. Bostoen, S. Mullender, and Y. Berbers, "Power-Reduction Techniques for Data-Center Storage Systems," *ACM Computing Surveys*, vol. 45, no. 3, pp. 1–38, 2013.

[23] A. Jahanshahi, H. Z. Sabzi, C. Lau, and D. Wong, "GPU-NEST: Characterizing Energy Efficiency of Multi-GPU Inference Servers," *IEEE Computer Architecture Letters*, vol. 19, no. 2, pp. 139–142, 2020.

[24] D. Zhao, S. Samsi, J. McDonald, B. Li, D. Bestor, M. Jones, D. Tiwari, and V. Gadepally, "Sustainable Supercomputing for AI: GPU Power Capping at HPC Scale," in *Proceedings of the 2023 ACM Symposium on Cloud Computing*, 2023, pp. 588–596.

[25] R. Desislavov, F. Martínez-Plumed, and J. Hernández-Orallo, "Trends in AI Inference Energy Consumption: Beyond the Performance-Vs-Parameter Laws of Deep Learning," *Sustainable Computing: Informatics and Systems*, vol. 38, p. 100857, 2023.

[26] V. Pallipadi and A. Starikovskiy, "The Ondemand Governor," in *Proceedings of the linux symposium*, vol. 2, no. 00216, 2006, pp. 215–230.

[27] D. Yokoyama, B. Schulze, F. Borges, and G. Mc Evoy, "The Survey on Arm Processors for HPC," *The Journal of Supercomputing*, vol. 75, pp. 7003–7036, 2019.

[28] B. M. Tudor and Y. M. Teo, "On Understanding the Energy Consumption of ARM-Based Multicore Servers," *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, no. 1, pp. 267–278, 2013.

[29] Q. Jiang, Y. C. Lee, and A. Y. Zomaya, "The Power of ARM64 in Public Clouds," in *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, 2020, pp. 459–468.

[30] M. Hennecke, M. Matsuda, and M. Nakao, "Evaluating DAOS Storage on ARM64 Clients," in *Proceedings of the HPC Asia 2023 Workshops*, 2023, pp. 65–72.

[31] A. Klimovic, C. Kozyrakis, E. Thereska, B. John, and S. Kumar, "Flash Storage Disaggregation," in *Proceedings of the Eleventh European Conference on Computer Systems*, 2016, pp. 1–15.

[32] L. Bindschaedler, A. Goel, and W. Zwaenepoel, "Hailstorm: Disaggregated Compute and Storage for Distributed LSM-based Databases,"

in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 301–316.

[33] "MLPerf Storage Benchmark Suite," https://github.com/mlcommons/storage, 2024, [Online; accessed 2-Jan-2024].

[34] Z. Guz, H. H. Li, A. Shayesteh, and V. Balakrishnan, "Performance Characterization of NVMe-over-Fabrics Storage Disaggregation," *ACM Transactions on Storage*, vol. 14, no. 4, pp. 31:1–31:18, 2018.

[35] D. Kumar and S. Li, "Separating Storage and Compute with the Databricks Lakehouse Platform," in *2022 IEEE 9th International Conference on Data Science and Advanced Analytics (DSAA)*, 2022, pp. 1–2.

[36] N. Li, A. Kalaba, M. J. Freedman, W. Lloyd, and A. Levy, "Speculative Recovery: Cheap, Highly Available Fault Tolerance with Disaggregated Storage," in *2022 USENIX Annual Technical Conference (USENIX ATC 22)*, 2022, pp. 271–286.

[37] E. Blem, J. Menon, T. Vijayaraghavan, and K. Sankaralingam, "ISA Wars: Understanding the Relevance of ISA being RISC or CISC to Performance, Power, and Energy on Modern Architectures," *ACM Transactions on Computer Systems*, vol. 33, no. 1, pp. 1–34, 2015.

[38] D. Patterson, "50 Years of Computer Architecture: From the Mainframe CPU to the Domain-Specific Tpu and the Open RISC-V Instruction Set," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, 2018, pp. 27–31.

[39] E. Blem, J. Menon, and K. Sankaralingam, "Power Struggles: Revisiting the RISC vs. CISC Debate on Contemporary ARM and X86 Architectures," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, 2013, pp. 1–12.

[40] T. L. Erxleben, K. Duwe, J. Saak, M. Kohler, and M. Kuhn, "Green Storage: Parallel File Systems on ARM," *International Journal on Advances in Software*, vol. 15, no. 3&4, pp. 200–210, 2022.

[41] V. Machado, A. Braga, N. Rampon, J. Bez, F. Boito, R. Kassick, E. Padoin, J. Diaz, J.-F. Méhaut, and P. Navaux, "Towards Energy-Efficient Storage Servers," in *Proceedings of the Symposium on Applied Computing*, 2017, pp. 1554–1559.

[42] P. J. Pavan, R. K. Lorenzoni, V. R. Machado, J. L. Bez, E. L. Padoin, F. Z. Boito, P. O. Navaux, and J. Méhaut, "Energy Efficiency and I/O Performance of Low-Power Architectures," *Concurrency and Computation: Practice and Experience*, vol. 31, no. 18, p. e4948, 2019.

[43] J. Xia, C. Cheng, X. Zhou, Y. Hu, and P. Chun, "Kunpeng 920: The First 7-nm Chiplet-Based 64-Core ARM SoC for Cloud Services," *IEEE Micro*, vol. 41, no. 5, pp. 67–75, 2021.

[44] "Performance Evaluation of the BeeGFS File System on the Arm AArch64 Architecture," https://thinkparq.com/wp-content/uploads/2019/08/White_Paper_BeeGFS_File_System-_on_the_Arm-_AArch64-_Architecture.pdf, 2019, [Online; accessed 7-Dec-2023].

[45] J. Maqbool, S. Oh, and G. C. Fox, "Evaluating ARM HPC Clusters for Scientific Workloads," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 17, pp. 5390–5410, 2015.

[46] R. Azimi, T. Fox, W. Gonzalez, and S. Reda, "Scale-Out vs Scale-Up: A Study of Arm-Based Socs on Server-Class Workloads," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, vol. 3, no. 4, pp. 1–23, 2018.

[47] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doylet, "Managing Energy and Server Resources in Hosting Centers," *SIGOPS Oper. Syst. Rev.*, vol. 35, no. 5, pp. 103–116, 2001.

[48] D. Narayanan, A. Donnelly, and A. Rowstron, "Write Off-Loading: Practical Power Management for Enterprise Storage," *ACM Transactions on Storage*, vol. 4, no. 3, pp. 10:1–10:23, 2008.

[49] A. Thomasian, "Saving Power in Disks, Flash Memories, and Servers," in *Storage Systems*, 2022, pp. 349–384.

[50] J. Yang, Y. Yue, and K. V. Rashmi, "A Large Scale Analysis of Hundreds of In-Memory Cache Clusters at Twitter," in *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, 2020, pp. 191–208.

[51] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, "Workload Analysis and Demand Prediction of Enterprise Data Center Applications," in *2007 IEEE 10th International Symposium on Workload Characterization*, 2007, pp. 171–180.

[52] K. Zhou, S. Sun, H. Wang, P. Huang, X. He, R. Lan, W. Li, W. Liu, and T. Yang, "Demystifying Cache Policies for Photo Stores at Scale: A Tencent Case Study," in *Proceedings of the 2018 International Conference on Supercomputing*, ser. ICS '18, 2018, pp. 284–294.

[53] S. Dong, S. S. P, S. Pan, A. Ananthabhotla, D. Ekambaram, A. Sharma, S. Dayal, N. V. Parikh, Y. Jin, A. Kim, S. Patil, J. Zhuang, S. Dunster, A. Mahajan, A. Chelluri, C. Datye, L. V. Santana, N. Garg, and O. Gawde, "Disaggregating RocksDB: A Production Experience," *Proceedings of the ACM on Management of Data*, vol. 1, no. 2, pp. 192:1–192:24, 2023.

[54] Z. Li, K. M. Greenan, A. W. Leung, and E. Zadok, "Power Consumption in Enterprise-Scale Backup Storage Systems," *Power*, vol. 2, no. 2, 2012.

[55] E. Rotem, A. Naveh, A. Ananthakrishnan, E. Weissmann, and D. Rajwan, "Power-Management Architecture of the Intel Microarchitecture Code-Named Sandy Bridge," *IEEE Micro*, vol. 32, no. 2, pp. 20–27, 2012.

[56] K. Ma and X. Wang, "PGCapping: Exploiting Power Gating for Power Capping and Core Lifetime Balancing in CMPs," in *Proceedings of the 21st international conference on Parallel architectures and compilation techniques*, 2012, pp. 13–22.

[57] "NVM Express. NVM Express Base Specification 2.0c. 2022." https://nvmexpress.org/specification/nvm-express-base-specification, 2022, [Online; accessed 7-Dec-2023].

[58] E. Asyabi, A. Bestavros, E. Sharafzadeh, and T. Zhu, "Peafowl: In-Application CPU Scheduling to Reduce Power Consumption of In-Memory Key-Value Stores," in *Proceedings of the 11th ACM Symposium on Cloud Computing*, 2020, pp. 150–164.

[59] P. Enberg, A. Rao, and S. Tarkoma, "I/O Is Faster Than the CPU: Let's Partition Resources and Eliminate (Most) OS Abstractions," in *Proceedings of the Workshop on Hot Topics in Operating Systems*, 2019, pp. 81–87.

[60] "CPU Hotplug in the Kernel," https://docs.kernel.org/core-api/cpu_hotplug.html, 2023, [Online; accessed 7-Dec-2023].

[61] J. Leverich, M. Monchiero, V. Talwar, P. Ranganathan, and C. Kozyrakis, "Power Management of Datacenter Workloads Using Per-Core Power Gating," *IEEE Computer Architecture Letters*, vol. 8, no. 2, pp. 48–51, 2009.

[62] E. L. Sueur and G. Heiser, "Dynamic Voltage and Frequency Scaling: The Laws of Diminishing Returns," in *Proceedings of the 2010 International Conference on Power Aware Computing and Systems*, 2010, pp. 1–8.

[63] W. Lin, F. Shi, W. Wu, K. Li, G. Wu, and A.-A. Mohammed, "A Taxonomy and Survey of Power Models and Power Modeling for Cloud Servers," *ACM Computing Surveys*, vol. 53, no. 5, pp. 100:1–100:41, 2020.

[64] H. Li, M. Hao, S. Novakovic, V. Gogte, S. Govindan, D. R. K. Ports, I. Zhang, R. Bianchini, H. S. Gunawi, and A. Badam, "LeapIO: Efficient and Portable Virtual NVMe Storage on ARM SoCs," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 591–605.

[65] H. Unnibhavi, D. Cerdeira, A. Barbalace, N. Santos, and P. Bhatotia, "Secure and Policy-Compliant Query Processing on Heterogeneous Computational Storage Architectures," in *Proceedings of the 2022 International Conference on Management of Data*, 2022, pp. 1462–1477.

[66] M. O'Connor, N. Chatterjee, D. Lee, J. Wilson, A. Agrawal, S. W. Keckler, and W. J. Dally, "Fine-Grained DRAM: Energy-Efficient DRAM for Extreme Bandwidth Systems," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, 2017, pp. 41–54.

[67] H. Hassan, M. Patel, J. S. Kim, A. G. Yaglikci, N. Vijaykumar, N. M. Ghiasi, S. Ghose, and O. Mutlu, "CROW: A Low-Cost Substrate for Improving DRAM Performance, Energy Efficiency, and Reliability," in *Proceedings of the 46th International Symposium on Computer Architecture*, 2019, pp. 129–142.

[68] G. F. Oliveira, S. Ghose, J. Gómez-Luna, A. Boroumand, A. Savery, S. Rao, S. Qazi, G. Grignou, R. Thakur, E. Shiu, and O. Mutlu, "Extending Memory Capacity in Modern Consumer Systems With Emerging Non-Volatile Memory: Experimental Analysis and Characterization Using the Intel Optane SSD," *IEEE Access*, vol. 11, pp. 105 843–105 871, 2023.

[69] E. L. Sueur and G. Heiser, "Slow Down or Sleep, That Is the Question," in *2011 USENIX Annual Technical Conference (USENIX ATC 11)*, 2011.

[70] J. C.-Y. Chou, T.-H. Lai, J. Kim, and D. Rotem, "Exploiting Replication for Energy-Aware Scheduling in Disk Storage Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 10, pp. 2734–2749, 2015.

[71] T.-Y. Chen, H.-W. Wei, T.-T. Yeh, T.-S. Hsu, and W.-K. Shih, "An Energy-Efficient and Reliable Storage Mechanism for Data-Intensive

Academic Archive Systems," *ACM Transactions on Storage*, vol. 11, no. 2, pp. 10:1–10:21, 2015.

[72] M. M. Al Assaf, X. Jiang, M. R. Abid, and X. Qin, "Eco-Storage: A Hybrid Storage System with Energy-Efficient Informed Prefetching," *Journal of Signal Processing Systems*, vol. 72, no. 3, pp. 165–180, 2013.

[73] D. Li, J. Wan, N. Zhao, D. Wen, C. Zhang, F. Wu, and C. Xie, "PreMatch: An Adaptive Cost-Effective Energy Scheduling System for Data Centers," in *International Conference on Massive Storage Systems and Technology*, 2020.

[74] S. Imamura, E. Yoshida, and K. Oe, "Reducing CPU Power Consumption with Device Utilization-Aware DVFS for Low-Latency SSDs," *IEICE Transactions on Information and Systems*, vol. E102.D, no. 9, pp. 1740–1749, 2019.

[75] B. Guo, J. Yu, D. Yang, H. Leng, and B. Liao, "Energy-Efficient Database Systems: A Systematic Survey," *ACM Computing Surveys*, vol. 55, no. 6, pp. 1–53, 2023.

[76] M. Poess and R. O. Nambiar, "Energy Cost, the Key Challenge of Today's Data Centers: A Power Consumption Analysis of TPC-C Results," *Proc. VLDB Endow.*, vol. 1, no. 2, pp. 1229–1240, 2008.

[77] Z. Xu, Y.-C. Tu, and X. Wang, "Online Energy Estimation of Relational Operations in Database Systems," *IEEE Transactions on Computers*, vol. 64, no. 11, pp. 3223–3236, 2015.

[78] M. Rodriguez-Martinez, H. Valdivia, J. Seguel, and M. Greer, "Estimating Power/Energy Consumption in Database Servers," *Procedia Computer Science*, vol. 6, pp. 112–117, 2011.

[79] C. Guo, J.-M. Pierson, H. Liu, and J. Song, "Frequency Selection Approach for Energy Aware Cloud Database," *IEEE Access*, vol. 7, pp. 1927–1942, 2019.

[80] M. Korkmaz, M. Karsten, K. Salem, and S. Salihoglu, "Workload-aware cpu performance scaling for transactional database systems," in *Proceedings of the 2018 International Conference on Management of Data*, 2018, pp. 291–306.