PCAP: Performance-Aware Power Capping for the Disk Drive in the Cloud

Mohammed G. Khatib and Zvonimir Bandic
WDC Research
{mohammed.khatib,zvonimir.bandic}@hgst.com

Abstract

Power efficiency is pressing in today's cloud systems. Datacenter architects are responding with various strategies, including capping the power available to computing systems. Throttling bandwidth has been proposed to cap the power usage of the disk drive. This work revisits throttling and addresses its shortcomings. We show that, contrary to the common belief, the disk's power usage does not always increase as the disk's throughput increases. Furthermore, throttling unnecessarily sacrifices I/O response times by idling the disk. We propose a technique that resizes the queues of the disk to cap its power. Resizing queues not only imposes no delays on servicing requests, but also enables performance differentiation.

We present the design and implementation of PCAP, an agile performance-aware power capping system for the disk drive. PCAP dynamically resizes the disk's queues to cap power. It operates in two performance-aware modes, throughput and tail-latency, making it viable for cloud systems with service-level differentiation. We evaluate PCAP for different workloads and disk drives. Our experiments show that PCAP reduces power by up to 22%. Further, under PCAP, 60% of the requests observe service times below 100 ms compared to just 10% under throttling. PCAP also reduces worst-case latency by 50% and increases throughput by 32% relative to throttling.

1 Introduction

The widespread adoption of on-line services has been fueling the demand for more and denser datacenters. The design of such warehouse-sized computing systems [12] is not at all trivial. Architects have to deal not only with computing, storage and networking equipment, but also with cooling and power infrastructures [13]. In fact, power and energy are first-order concerns for architects as new high-performing hardware is likely to require more power, while the cost of hardware has remained stable. With these trends continuing, Barroso [11] argues that the cost of the energy to operate a server during its lifetime could surpass the cost of the hardware itself.

Power is more concerning since the cost of building a datacenter is mainly dictated by the costs of its power infrastructure. These costs typically range between $10 and $20 per deployed Watt of peak critical power [29]. Hence, a datacenter with a provisioned 10 MW peak power capacity costs $100M to $200M (excluding cooling and ancillary costs), a significant amount of money. Contrast the $10/W building cost with an average of $0.80/Watt-hour for electricity in the U.S. Still, while energy costs vary with the usage of the datacenter, building costs are fixed and based on the *peak* power capacity. Consequently, it becomes very important to fully utilize a datacenter's power capacity. If a facility is operated at 85% of its maximum capacity, the cost of building the facility surpasses the energy costs for *ten years* of operation [29].

Recent studies have addressed maximizing the power utilization in datacenters [26, 36]. Researchers have characterized the power usage at different levels in the datacenter (e.g., machine and cluster) and investigated power provisioning strategies. One especially promising strategy is *power over-subscription* [12], that oversubscribes a datacenter with more machines (and thus more work) to ensure near 100% power utilization. The incentive to fully utilize the power budget is, however, offset by the risk of overloading the power trains and infrastructure of the datacenter. Such overloading could result in long service downtimes (due to power outages) and/or costly contractual fines (due to service agreement violations). To prevent overloading, power capping techniques are deployed as a safety mechanism, thus allowing maximum utilization while preventing costly penalties.

Power capping is a mechanism that ensures that the power drawn by a datacenter stays below a predefined

power limit or cap. At the core of power capping is a monitoring loop, which takes in power readings, and computes the amount of power capping needed. Capping itself is done in a variety of techniques depending on the scale and type of the hardware component under question. On a datacenter level, capping is an aggregate number that trickles down to clusters, racks, machines and components. Suspending low-priority tasks in a cluster and adapting the clock frequency of a CPU component are two example techniques.

This work focuses on capping the power usage of the storage component of the datacenter. We address the 3.5-inch high-capacity enterprise hard disk drives (HDDs) common in today's cloud deployments. This paper tackles the question of: *How can the* HDD *power consumption be capped in a performance-aware manner?*

To this end, we revisit the throttling technique proposed for power capping [40] and address its shortcomings in a new technique we propose. We show that throttling *unnecessarily* sacrifices timing performance to cap power. Throttling limits disk throughput by *stopping servicing and delaying all outstanding requests*. It is predicated on the assumption that low throughputs result in less power usage by the disk. Our power measurements reveal that, contrary to the common belief, the power usage of the disk does not always increase with the increase in throughput but declines for high throughputs. We find *no strict positive correlation between the power usage and the throughput of the disk.*

To enable power capping for the disk drive, we propose a technique that resizes the I/O queues. We show that resizing queues not only reduces the impact on performance, unlike throttling, but also enables two different performance-oriented operation modes: throughput and tail-latency. This is important given the various services offered by today's datacenters. For instance, web search is sensitive to latency, whereas Map-reduce is throughput-oriented [22, 35]. By I/O queues we mean both the disk's queue as well as its respective OS queue. We investigate the interplay between both queues and their influence on the disk's power usage.

We present PCAP, an agile performance-aware power capping system for the disk drive. PCAP dynamically adapts the queues of a disk drive to cap its power. It performs power capping in two different operation modes: throughput and tail-latency. Under PCAP, 60% of the requests exhibit latencies less than 100 ms as opposed to just 10% under throttling. Also, PCAP reduces worst-case latency by 50% and increases throughput by 32% compared to throttling. PCAP has three goals:

1. Agile power capping that reacts quickly to workload variations to prevent power overshooting as well as performance degradation.

2. Graceful power control to prevent oscillations in power and better adhere to service level agreements.

3. Maximized disk performance for enhanced performance of applications.

This paper makes the following contributions:

- Revisiting the throttling technique for HDDs and studying the throughput–power relationship (section 4).
- Investigating the impact of the HDD's and OS queues on the HDD's power and performance (section 5).
- Designing and evaluating the PCAP system that is agile, graceful, and performance-aware (section 6).

This paper is structured as follows. The next section offers a brief refresher of the basics of HDDs. Section 3 evaluates the merit of power capping for HDDs and presents our experimental setup. Section 4 revisits throttling and its impact on power. Section 5 investigates the influence of the queue size on HDD's power consumption. Section 6 presents the design of PCAP and Section 7 evaluates it. Section 8 studies PCAP for different workloads. Section 9 discusses related work and Section 10 concludes.

2 Background

2.1 Power capping vs. Energy efficiency

This work focuses on power capping to maximize the utilization in the datacenter, where peak power predominates costs of ownership. We do not address energy efficiency, where machines are powered down in underutilized datacenters to save energy. While the latter received ample attention in the literature [43, 44, 45, 48, 41], the former received relatively little [40, 26, 36].

Addressing power capping, we measure power dissipation. Power is the rate at which electricity is consumed. It is measured at an instant in time as Watts (W). To contrast, energy is a total quantity and is power integrated over time. It is measured as Wh (Watt-hour), or joules. We focus on power usage here.

2.2 HDD power capping

The active read/write mode of the HDD is of a primary interest for power capping. This is because the HDD draws most of the power in the active mode (e.g., compare 11 W during activity to 6 W during idleness). Also, in cloud systems, HDDs spend most of the time in the active mode [17]. Generally speaking, power capping may transition the HDD between the active mode and one or more of its low power modes to reduce the average power drawn in a period of time. Throttling for instance transitions the disk between the active and idle modes. This transitioning comes at a performance penalty, which scales with the depth and frequency the low-power mode

being visited. In contrast, in this work we avoid transitioning between power modes and propose the adjustment of the queue size to achieve power capping for the disk drive in the active mode.

2.3 HDD's IOQ and NCQ queues

Any block storage device, that is managed by an Operating System (OS), has a respective queue as a part of the OS [14]. The queue serves as space for the I/O scheduler to reorder I/O requests for increased throughputs. For example, the Linux OS maintains a queue depth of 128 requests by default (in the current Ubuntu distributions). Requests are reordered to optimize for sequentiality on the HDD. The queue size is adjustable with a minimum size of 4. We refer to this queue as IOQ in this work.

NCQ stands for Native Command Queuing [5]. It is an extension to the Serial ATA protocol that allows the HDD to internally optimize the order in which requests are serviced. For instance, the HDD may reorder requests depending on its rotational positioning in order to serve all of them with fewer rotations and thus less time. NCQ typically ranges from 1 to 32, where NCQ=1 means disabled NCQ.

2.4 Scope of this work

We focus on the storage component of the datacenter. Making storage power-aware enables better overall power provisioning and capping. The share in power consumption due to storage varies. For example, in storage-heavy systems, such as the HGST Active Archive System [3], 80% of the power is due to the 600 HDDs it hosts, whereas in a computing-heavy system, such as the Dell PowerEdge R720 Server, 5-10%. We propose a technique to cap power at the disk level. Other techniques exist that may operate at different levels. We envision our technique complementing other techniques to achieve datacenter-wide power capping. Our technique has potentially wide applicability, since it (1) has no influence on data availability, (2) works under heavy workloads (i.e., no idle periods), (3) has no impact on HDD reliability, and (4) enables fine-tuned Watt-order capping. It offers three key properties: sensitive to performance, non-invasive to the I/O software stack, and simple to understand and implement (see Section 6).

3 The case for HDD power capping

In this section, we address a merit question: *How much power can be saved by power capping the* HDD*?*

To this end, we quantify the range of power an HDD draws when servicing requests, called dynamic power. We discuss the setup we prepared for our studies first.

3.1 Hardware setup

We chose a JBOD setup (Just a Bunch of Disks) to host a set of HDDs, which are exercised and their power is measured. The JBOD setup consists of a Dell PowerEdge R720 Server connected via LSI 9207-8e HBA to a Supermicro JBOD. It holds 16 3.5" SATA HDDs. We selected HGST Ultrastar 7K4000 of 4 TB capacity [2], commonly found in cloud storage systems today.

Besides the HGST Ultrastar 7K4000, we have obtained a sample HDD from two other HDD vendors. We selected a Seagate 4TB HDD [6] and a WD 4TB HDD [8]. All disks have the same capacity and number of platters, since they share the same storage density (i.e., same generation). We use different disks to ensure the commonality of our observations as well as the applicability of our techniques across different vendors, generations and technologies.

We profile power using WattsUp .NET power meters [7]. We use one meter for the JBOD and another for the server. We interpose between the power supply and the mains. Since the JBOD is dual corded for high availability, we connect both to an electric strip which in turn goes into the power meter. The meters are connected via USB to the server, on which we collect power read-outs for later analysis. The meters sample power once per second. This rate should be enough for power capping, since capping is performed on higher time scales [12].

3.2 Software

Our server runs a 64-bit 12.02 Ubuntu Server distribution with the 3.0.8 Linux kernel. No OS data were stored on the disks in the JBODs. Instead, we used a separate disk for that purpose, which is housed in the server itself. Unless pointed out otherwise, the default settings of the I/O stack were kept intact. For example, the file systems on our disks (`XFS` in this case) were formatted with the default settings. The I/O scheduler was kept at the `deadline` default scheduler.

We used existing Linux utilities and implemented our own when needed. Utilities were used to generate workloads and collect logs of timing performance and power. We installed a WattsUp utility that communicates with the power meter and logs power read-outs. As for benchmarking, we use the FIO benchmark [10] to generate different workloads. We use FIO for design space exploration. To generate real workloads, we use MongoDB [4]. Because the usage of a benchmark varies depending on the goal of the experiment, we defer talking about the setup to each individual discussion of our studies. For timing performance profiling, we use the `iostat` Linux utility and benchmark-specific statistics. The collected performance and power logs are then fed

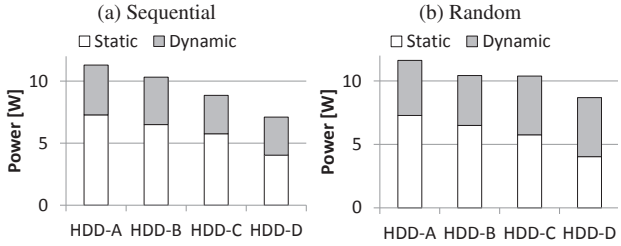

Figure 1: The dynamic and static power components measured for the four sample HDDs

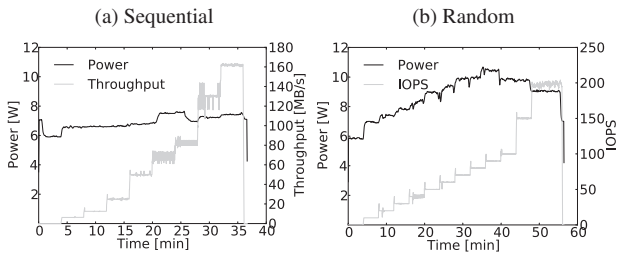

Figure 2: The power–throughput relationship under sequential and random workloads

to a custom-built Python program for analysis. Unless otherwise mentioned, we always collect measurements on the application level for end-to-end performance.

3.3 HDD's dynamic power

We measured the dynamic power for the four different types of HDD we have; the static power was isolated in separate measurements with no I/O load. Using FIO, we generated sequential and random workloads to measure the maximum power for each individual HDD. Under sequential workloads, the maximum power corresponds to the maximum attainable throughput, which is approximately 170 MB/s. In contrast, the maximum power under random workloads is attained by ensuring maximum seek distance between consecutive requests (Section 4). Figure 1 shows HDD's power broken down into static and dynamic components. The static component is related to the spindle and the electronics, whereas the dynamic component is related to the head arm and read/write channel. Across all HDDs, the figure shows that the dynamic power ranges up to 4 W and 5 W under sequential and random workloads, respectively. Hence, for our JBOD system, which can hold up to 24 HDDs, power capping exhibits a range up to 96 W and 120 W, respectively. As such, *HDDs enjoy a sizable dynamic power range for power capping to conserve power.*

4 HDD's throughput throttling

Throttling has been proposed as a technique to cap HDD's power [40]. This section investigates the relationship between the power and throughput in HDDs. We implemented a kernel module that enables us to throttle throughput under sequential as well as random workloads, called `dm-throttle`. The module is based on the device-mapper layer of the Linux kernel and is 700 lines of C code. It accepts as an input the desired throughput cap in KB per second or IOs per second for sequential and random workloads, respectively. The throughput cap can be modified at run-time via the `/proc/` file system, where statistics are accessed too.

We set up two instances of FIO to generate sequential and random workloads for 40 and 50 minutes, respectively. We used multiple threads in the random workload generation to attain maximum disk throughput. During the run of each FIO instance, we varied the throttling throughput of `dm-throttle` and measured the power and the effective throughput. Throughput was throttled at several points (6.25, 12.5, 25, 50, 75, 100, 150, 200 MB/s) for sequential workloads, and (10−100, 150, 200 IOPS) for random workloads. In these experiments, we used one HDD from the 16 (identical) HDDs. We singled out its power after subtracting the idle power of the JBOD (incl. power supplies, fans and adapters).

Figures 2a and 2b show the throughput–power relationship under sequential and random workloads, respectively. Figure 2a shows that the HDD draws more power (the black curve) as throughput increases (the gray curve). The HDD draws 8 W of power at the maximum throughput (170 MB/s). And its power range, that scales with the throughput, is $7-8$ W. Another 0.6 W is added to the dynamic range due to channel activity when exercising the disk with some workload. This effect can be seen in Figure 2a during the second four minutes of the experiment. In separate measurements, we found that an additional 1 W of power is drained when the heads are loaded to counteract the drag. As such, the total dynamic range for such disks is $5.5-8$ W, which is in agreement with the figures for HDD-C in Figure 1a.

Figure 2b shows that under random workloads power increases with the throughput up to a certain point, 90 IOPS in this case. After that, power starts decreasing for higher throughputs with a noticeable drop at the maximum throughput of 200 IOPS, thanks to better scheduling in the disk (see the next section). The figure highlights the fact that different throughputs can be attained for the same amount of power drawn. Also, the dynamic power range is wider under random workloads compared to sequential workloads, between $7-10.5$ W versus $7-8$ W (excluding the power due to channel activity and head positioning). As such the *effective* power ranges for our JBOD are up to 24 W and 84 W under sequential and random workloads, respectively.

Summarizing, we make two key observations that guide the rest of this work:

Observation 1: *HDDs exhibit a dynamic power range that is wider under random workloads compared to sequential workloads.*

Observation 2: *HDDs can offer different throughputs and service times for an equivalent amount of power under random workloads.*

The two observations lead us to investigate a power-capping technique under random workloads (Observation 1) that is performance-aware (Observation 2). We focus on random workloads in the rest of this paper. If required, power-capping by throttling should be a sufficient technique for sequential workloads, thanks to the positive correlation between power and throughput *under sequential workloads*. Next, we investigate the reason for the decline in the HDD's power consumption at high throughputs, which motivates the choice for using the queue size to control power.

5 Power–Queue relationship

This section explains the dynamics of the HDD's head arm. We motivate our choice for the queue size to control power. We then investigate the influence of the queue size on the HDD's power, throughput, and service time.

5.1 Causality

Under random workloads, the HDD's head moves across the surface of the platters to service requests from different locations. The head motion is characterized as random, since the head spends most of the time seeking instead of accessing bits (compare 5 ms seek time to 0.04 ms 4 KB read time). Moving the head dissipates power by its VCM (voice-coil motor). Depending on the physical distance separating any two consecutive requests, the head may need to accelerate and subsequently decelerate. Acceleration and deceleration require a relatively large amount of power (similar to accelerating a car from standstill). *A few (random) requests have a long physical distance in between, requiring acceleration and deceleration. Conversely, the more the requests dispatched to the disk, the shorter the separating distance and thus the less the power due to reduced acceleration, if any.* At higher loads more requests are dispatched to the disk simultaneously, allowing the disk to better schedule and reduce distances and thus accelerations resulting in less power. In Figure 2b, the disk consumes less power at low throughputs ($<$ 90 IOPS) too but for a different reason. At low throughputs, the disk is underutilized and spends more than 45% of the time in the idle power mode, resulting in power savings that outweigh the increase in power due to (occasional) accelerations.

5.2 Characterizing the relationship

This section studies both the IOQ (scheduler) queue and the NCQ queue described in Section 2.3. We investigate their interplay and influence on power and performance. We seek to answer the following two questions:

1. For a fixed NCQ queue size, what is the relationship between the IOQ queue size and the HDD's power, throughput and service time?

2. For a fixed IOQ queue size, what is the relationship between the NCQ queue size and the HDD's power, throughput and service time?

Methodology We studied a single HDD in our JBOD system from Section 3.1. We carried out two sets of experiments to confirm trends: once with FIO and another with MongoDB [4]. We generated random 4KB requests with FIO using `aiolib`. We used enough threads to mimic real systems with multiple outstanding requests. For the MongoDB setup, we stored 100 databases on the disk, each of which is approximately 10 GB in size. The files of every two consecutive databases (e.g., 1-st and 2-nd) were separated by a 10-GB dummy file on the disk, so that 2.4 TB was consumed from our 4 TB HDD. The disk was formatted with the `XFS` file system using the default settings. We used YCSB [19] to exercise 10 databases, the 10-th, 20-th, 30-th, up to the 100-th. One YCSB instance of 10 threads was used per database to increase throughput. We benchmarked for different queue sizes. The IOQ queue was varied in the range (4, 8, 16, 32, 64, 128), whereas the range for the NCQ queue was (1, 2, 4, 8, 16, 32). To resize a queue to a value, say SIZE, we used the following commands:

- IOQ queue: `echo SIZE > /sys/block/sdc/queue/nr_requests`
- NCQ queue: `hdparm -Q SIZE /dev/sdc`

Results Figure 3a shows HDD's power versus the size of the IOQ queue. Power decreases as the IOQ queue size increases. A large IOQ queue enables better scheduling and thus reduces randomness in requests arriving to the disk (which in turn reduces accelerations). A trend exists where power reduction exhibits diminishing returns at large queues, since only the power due to the head's VCM is affected, whereas other static power components remain intact (Amdahl's law). The figure confirms the same trend for different sizes of the NCQ queue, but at different power levels.

Figure 3b shows HDD's power versus the size of the NCQ queue. Unlike for the IOQ queue, two opposing trends exist. In fact, the size of the IOQ queue plays a major role here. We can explain the figure by observing trends at small and large IOQ queue sizes (e.g., 4 and

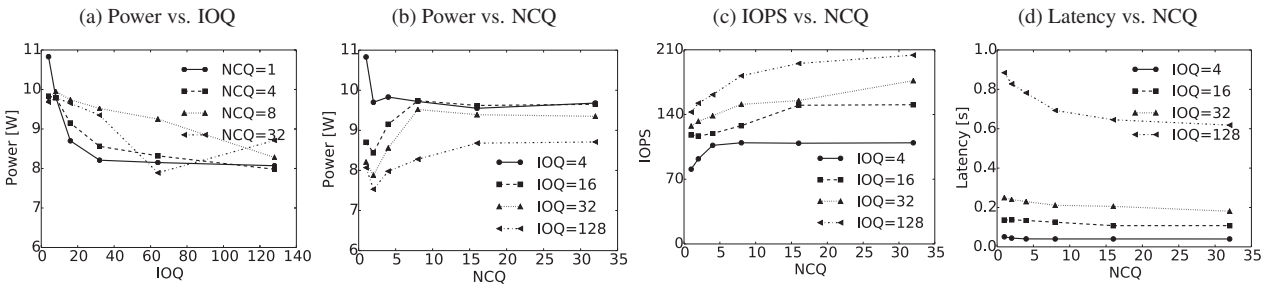

Figure 3: The influence of the queue size for both IOQ and NCQ on the HDD's power, throughput and service time

32, respectively). At small sizes, power decreases as the NCQ queue size increases, because the requests arriving at the disk sill exhibit randomness, leaving room for better scheduling by NCQ. Recall that better scheduling reduces acceleration and thus leads to lower power consumption. Conversely, at large sizes of the IOQ queue, power increases as the NCQ queue size increases, since randomness is already reduced by the I/O scheduler and thus even higher throughputs are attained at large NCQ queue sizes (200 IOPS versus 100 IOPS on the bottom curve of Figure 3c). High throughputs involve more channel activity, which draws more power.

As for the timing performance, Figure 3c shows the relationship between throughput, in IOPS, and the NCQ queue size. Expectedly, throughput increases at large queues, since more time is spent on accessing bits rather than seeking. We observed similar trends for throughput versus the IOQ queue. We do not show it for space reasons. One observation is that the HDD's throughput is mainly limited by the IOQ size. That is, increasing NCQ beyond IOQ size does not result in increased throughput, since NCQ scheduling is limited by the number of requests it sees at a time, which is in turn bounded by the IOQ size.

Figure 3d shows the relationship between the HDD service time, measured by `iostat`, and the NCQ queue size. Surprisingly, the service time decreases for large NCQ queue sizes, although larger queuing delays are incurred. This suggests that the saving in rotational positioning time due to NCQ scheduling outweighs the queuing delay of large queues. This improvement is more pronounced for large numbers of arriving requests as shown by the top curve in the figure for an IOQ size of 128. Conversely but expectedly, we observed a linear increase in service time as the IOQ queue size increases. We do not show it for space reasons.

Summarizing, HDD power decreases with the increase in the size of the IOQ (scheduler) queue. Both throughput and service time expectedly increase. On the other hand, while throughput increases with the increase of the NCQ queue size, power and service time have unusual trends. We make two new observations:

Observation 3: *The power drawn by the HDD exhibits opposing trends with respect to the NCQ queue size. These trends are influenced by the size of the IOQ scheduler queue.*

Observation 4: *The HDD's service time decreases with the increase in the size of the NCQ queue, thanks to improved in-disk scheduling.*

The interplay between the two queues leads to the following observation:

Observation 5: *The HDD's dynamic power range can be fully navigated by varying the sizes of the NCQ and I/O scheduler queues.*

6 PCAP design

In a dynamic datacenter environment, where power requirements and workloads change constantly, a control system is required. Such a system ensures compliance to power caps when power is constrained and enables better performance when more power is available. We present the design of PCAP and techniques to make it graceful, agile and performance-aware. PCAP's design is based on the observations made previously.

6.1 Base design

At its core, PCAP has a control loop that triggers every period, T. It computes the running average of the power readings over the past T time units and decides on the amount of power capping needed. PCAP is a proportional controller. To attain better performance and ensure stability, we improve upon the basic proportional controller in four ways. (1) PCAP increases and decreases power using models derived from the observations of Section 5. (2) It uses different gains when increasing and decreasing power. (3) PCAP bounds the ranges of its control signals (i.e., queue sizes) and (4) employs a hysteresis around its target output to prevent oscillations due to measurement errors. Power is increased or decreased by adjusting the size of the IOQ and NCQ queues gradually, one step per period. Queue adjustment is based on the relationships investigated in Section 5.2. PCAP uses two factors, α_{up} and α_{dn}, to increase power (or allow

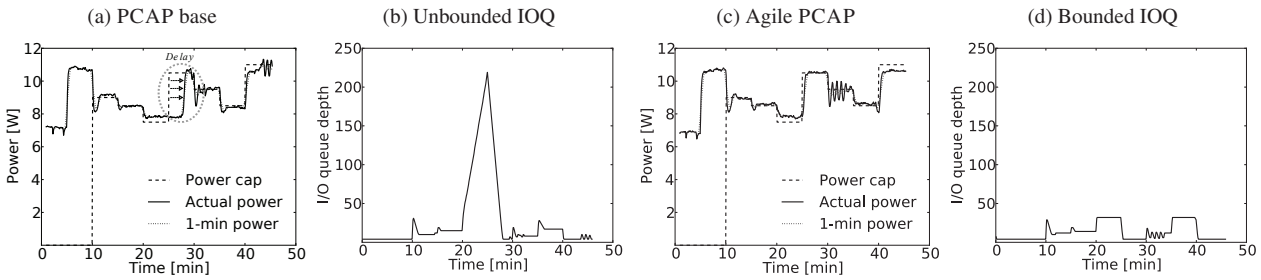

Figure 4: Power capping with PCAP and the corresponding queue sizes for the (a & b) base and (c & d) agile designs

better performance) and to decrease power, respectively. We use two factors, since the decrease in power must be done aggressively to avoid penalties, whereas the increase in power is done incrementally and cautiously. Consequently, α_{dn} is greater than α_{up}. Based on either factor, the queue size is adjusted proportionally to how far the target power, P_t, is from the current power, P_c. We use the following equations to calculate the change in the queue size, ΔQ, to enable *graceful control*:

$$\Delta Q_{IOQ} = \frac{|P_t - P_c|}{\Delta P_{IOQ}} \cdot 2\alpha_{dn} \quad (1)$$

$$\Delta Q_{NCQ} = \frac{|P_t - P_c|}{\Delta P_{NCQ}} \cdot \alpha_{dn} \quad (2)$$

ΔP_{IOQ} and ΔP_{NCQ} are the maximum change in power attainable by varying the IOQ and NCQ queues, respectively. We multiply α_{dn} by 2 for the IOQ to attain measurable changes in power. Changing queue sizes to allow for power increase follows Equations 1 and 2 after replacing α_{dn} with α_{up}. To account for measurement errors and reduce oscillations, an error margin of ε is tolerated around the target power. That is no further power-capping actions are taken, if the current power is within a range of [-ε, +ε] of the target power. For our experiments, we settled on the settings for PCAP's parameters shown in Table 1. These are, however, configurable depending on the conditions of the datacenter as well as its operation goals.

We implemented a prototype of PCAP in Python. It is 300 lines of code. PCAP runs in the background. New power targets, if any, are echoed into a text file which PCAP reads in at the beginning of every period, T. After that, it executes the control loop explained before until the target power is attained. Figure 4a shows the activity of PCAP on a single HDD over a 45-minute period of time. The figure shows three curves, the target power cap (dashed), the instantaneous power (solid), and the 1-min average power (dotted). We generate a random workload for the entire period. Initially, we leave the disk idle for 5 minutes and then generate a workload with no power-capping for another 5 minutes. The HDD draws approximately 8 W and 11 W, respectively. At minute 10, the power cap is set to 9 W and PCAP adjusts queues to reduce HDD's power by 2 W to below 9 W. At minute 15, the power cap is again lowered to 8.5 W and PCAP lowers power accordingly. At minute 20, PCAP is unable to lower the power below 7.5 W, since it is outside the dynamic range of the HDD. We keep capping power at different levels for the rest of the experiment and PCAP reacts accordingly. Figure 4b shows the change in the queue sizes to attain power-capping.

The figure shows a delay at minute 25 in responding to raising the power cap from 7.5 W to 10.5 W. We study this sluggishness in the next section. Datacenter operators may by more interested in long-term smoothed averages for which contractual agreements may be in place. For example, the 1-minute power curve in the figure inhibits oscillations, unlike the instantaneous power curve, so that power violations should be of no concern in this case. We discuss performance in Section 6.3.

Table 1: PCAP's parameters and their settings

Parameter	Setting	Description
T	5 s	control loop period
α_{up}	2	factor used for power increase
α_{dn}	8	factor used for power decrease
ε	0.2 W	control tolerance factor
ΔP_{IOQ}	2 W	max power change with IOQ queue
ΔP_{NCQ}	2 W	max power change with NCQ queue
$[Q^L_{IOQ}, Q^U_{IOQ}]$	$[4, 32]$	IOQ queue range
$[Q^L_{NCQ}, Q^U_{NCQ}]$	$[1, 8]$	NCQ queue range
Q^p_{IOQ}	128	IOQ setting for maximizing throughput
Q^p_{NCQ}	32	NCQ setting for maximizing throughput

6.2 Agile capping

The oval in Figure 4a highlights an inefficiency in the base design of PCAP. It manifests as delays in increasing power when the power cap is lifted up. This inefficiency results in low throughputs and long service times, since queues take some time to adjust accordingly as shown at minute 25, calling for better agility.

We examined the cause of the delay in adjusting queue sizes. Figure 4b shows that the IOQ queue reached high sizes during the tracking of the previous very low power target (7.5 W). This is because the base design keeps in-

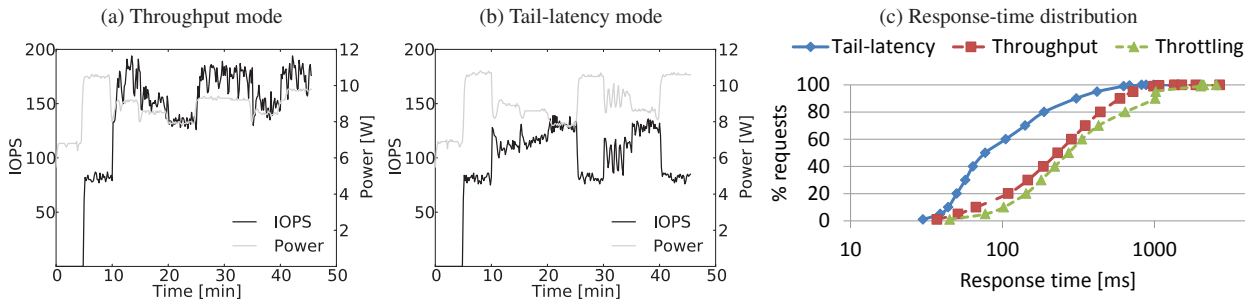

Figure 5: Throughput and service time under the throughput and tail-latency performance modes of PCAP

creasing the size of the IOQ queue until the power target is reached. In this particular case, IOQ queue size reached 219. Similarly, the NCQ queue can increase significantly. Later, when new and higher power targets are set as in minute 25, PCAP takes long time to reduce the queue size to low values since this happens gradually. The sluggishness results in a relatively long time period of lost performance.

To improve agility, we leverage an observation made from Figure 3a namely, power exhibits diminishing returns at high queue sizes, so that their respective power savings are marginal. As such, we introduce upper bounds on the sizes of both queues, Q^U_{IOQ} and Q^U_{NCQ}. The bounds limit the increase in queue sizes and enable shorter times to navigate queue ranges. Similarly, we introduce lower bounds. Figure 4c shows the performance of PCAP with bounds. Thanks to its queue-depth bounding, PCAP avoids infinitely and hopelessly attempting to cap power. Figure 4d confirms that the queue sizes never exceed the bounds. The bounds are shown in Table 1 and were set with values of the knees of the curves of Figures 3a and 3b. Figure 4c confirms *PCAP's agility*.

6.3 Performance awareness

Resizing queues impacts the performance of the HDD. We distinguish between two types of timing performance: (1) throughput and (2) tail-latency. In the throughput mode, PCAP attempts to increase throughput while adhering to the power cap. This mode enhances the average latency. In the tail-latency mode, PCAP attempts to reduce the high-percentiles latencies or alternatively shorten the tail of the latency distribution. In practice, the designer can set performance targets along the power target to reach compromises between the two.

As discussed in Section 5.2, throughput increases by increasing the size of both IOQ and NCQ queues. Power decreases with the increase in the IOQ queue size, whereas it increases for large NCQ queue sizes as shown in Figure 3c. PCAP uses models of these relationships to increase throughput while capping power. In contrast, the tail-latency decreases (i.e., high-percentile latencies decrease) for small IOQ queue sizes and large NCQ queues as shown in Figure 3d. We also incorporate models of this relationships in PCAP to reduce tail latencies while capping power.

The solution for agility of the previous section is in conflict with maximizing throughput (in PCAP's throughput mode). This is because the low upper bound on the size of both queues limits the maximum attained throughput. Compare 150 IOPS at (NCQ=8, IOQ=32) to 200 IOPS at (NCQ=32, IOQ=128) in Figure 3c, a 25% potential loss in throughput. To prevent this loss, we redesigned PCAP such that when it reaches the "agility" upper bounds, it snaps to predefined queue settings to maximize throughput in the throughput mode. The corresponding queue parameters are Q^ρ_{IOQ} and Q^ρ_{NCQ} (see Table 1). Similarly, PCAP snaps back from these settings to the upper bounds in the downtrend. That way, agility is still preserved while throughput is maximized. This scheme has no effect in the tail-latency mode, since small queues are required.

Figures 5a and 5b show the throughput of a single HDD when running PCAP in the throughput and tail-latency modes, respectively. Throughputs up to 200 IOPS are attained in the throughput mode, which is higher than the 140-IOPS maximum throughput attained in the tail-latency mode. The high throughput comes at the cost of long response times, resulting in a long tail in the distribution of the response time as Figure 5c shows. The figure plots the corresponding cumulative distribution function for the response time measured at the client side for both PCAP's modes. Tail-latency attains shorter maximum latencies, compare 1.3 s to 2.7 s maximum latency. Further, 60% of the requests observe latencies shorter than 100 ms in the tail-latency mode as opposed to just 20% in the throughput mode. We discuss the curve corresponding to throttling in Section 7.1.

7 PCAP's performance

This section compares PCAP to throttling and then studies the influence on I/O concurrency on PCAP's ability to cap power.

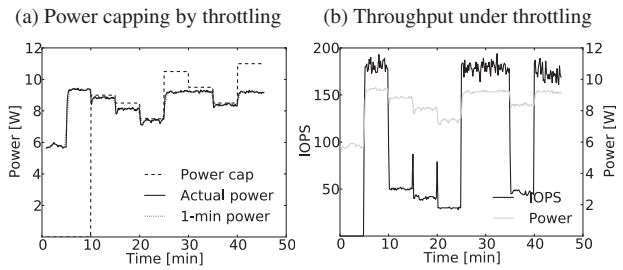

Figure 6: Using throttling to cap (a) HDD's power usage, while (b) maximizing throughput

7.1 PCAP versus throttling

Both queue resizing and throughput throttling can be used for capping power as evidenced in Figures 5a and 2a, respectively. Section 5.1 presented a qualitative argument as for why to use queue resizing to control power. This section presents a quantitative argument. We compare the timing performance of the HDD when its power is capped by either queue resizing (i.e., PCAP) or throttling. Precisely, we answer the question of: *For the same power cap, how do the throughput and the service time compare under throttling and* PCAP*?*

We repeated the same experiments of Section 6.3 while throttling the throughput (see Figure 2a) to cap the power within the desired levels. We used our `dm-throttle` module for throttling to reproduce Figure 4c. To take advantage of the HDD's low power consumption at high throughputs (Figure 2a), we used throttling only for power caps strictly below 9.5 W (which corresponds to the power drained at maximum throughputs). For higher power caps, we disabled throttling to maximize the performance of the disk. As such, `dm-throttle` was used to cap power selectively while avoiding hurting performance whenever possible, hence offering the best-case performance of throttling.

Figure 6a shows the power consumption of the HDD. Throttling keeps the power below the power cap, which is shown in dashed black. The power curve separates from the power cap by a large margin during some periods, such as minutes 25–30 and 40–45. These periods correspond to power capping with no throttling deployed, since the power cap is above 9.5 W. Figure 6b shows the throughput of the disk over the time period of the experiment. It confirms maximum throughputs around 185 IOPS during periods of no throttling. Comparing Figure 6b to Figures 5a and 5b we can visually see that throttling attains lower throughputs than the throughput mode of PCAP, whereas it outperforms the tail-latency mode of PCAP. While throttling attains an average of 117 IOPS, PCAP attains 154 IOPS (32% more) and 102 IOPS (15% less) in the throughput and tail-latency modes, respectively. Figure 5c shows the cumulative distribution of the response time for the requests of the previous experiment. Overall, throttling attains worse latency than the two performance modes of PCAP. Just 10% of the requests exhibit latencies under 100 ms, whereas the maximum latency is 2.5 s. This is however expected, since throttling delays requests and maintains default queue settings that are deep and optimized towards throughput. By treating response time and throughput differently, PCAP outperforms throttling on both performance goals. 60% of the requests exhibit response times below 100 ms and an increase of 32% in throughput is attainable with PCAP.

PCAP relies on concurrency in I/O requests to attain capping as we shall see next.

7.2 Influence of concurrency on PCAP

We carried out two sets of experiments to study PCAP's performance on a single HDD. In the first set, we varied the number of concurrent threads, while maximizing the load per thread so that the HDD utilization is 100% all the time. In the second set, we fixed the number of threads and varied the load per thread to attain different utilization levels.

Our goal is to study the relationship between the effective queue size (i.e., the actual number of outstanding requests at any time instance) and PCAP ability to cap the HDD's power. In the first experiment, since utilization is maximized the effective queue size matches the number of threads. In the second experiment, the effective queue size varies, since the load varies.

Maximized load In the first experiment, we varied the number of threads in the range (1, 2, 4, 8, 16, 32, 64, 128). We ran PCAP in the throughput mode and in the tail-latency mode, and chose different power caps. Figure 7a shows the average power measured over the period of the experiment for each setting. The figure shows that for a few threads the average power is 10.7 W, far above the power targets (8 W and 9 W). That is PCAP cannot cap power at concurrency levels below 4 threads (i.e., queue sizes under 4), since little opportunity exists for reordering requests and saving power. At concurrency levels above 4, the two curves start declining and departing from each other. This signifies that PCAP starts achieving some capping but cannot attain the target cap (i.e., no perfect capping). Finally, it attains the 9 W cap at 32 threads, whereas the 8 W cap is attained at 64.

We repeated the same experiment while running PCAP in the tail-latency mode. We studied for three power targets including 10 W, since the HDD enjoys larger dynamic range for small queues. We observed the same trend of attaining capping at higher concurrency levels. The power caps were attained at smaller number of threads compared to the case with the throughput

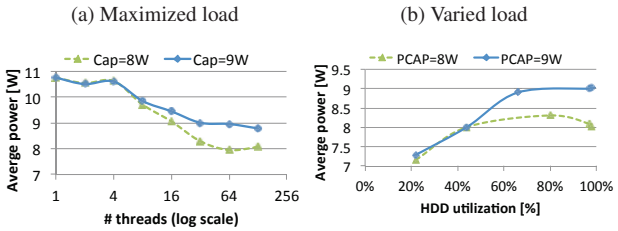

Figure 7: The influence of I/O concurrency on PCAP's performance in capping power for a single HDD

mode. For example, 9 W and 8 W were attained with as little as 16 (vs. 32) and 32 (vs. 64) threads, respectively. In addition, for the higher power target of 10 W, PCAP attains the cap starting from 2 threads.

Varied load In this experiment, we pushed load to the HDD progressively while fixing the number of threads. We used 32 threads to ensure power capping with PCAP. Without capping, the load varies between 32 IOPS to 200 IOPS. Figure 7b plots the average measured power versus the utilization level reported by `iostat`. The figure shows that the power is lower than both targets at low utilization, since the HDD spends a sizable fraction of time idling, thanks to the low load. Observe in such a scenario, throttling is happening "naturally". At higher utilizations, PCAP perfectly caps the power to below 9 W, whereas the 8 W cap is violated for utilization levels between 40–95%. To explain these results, we examined the effective queue size (from `iostat`) versus utilization. We found that the overshooting over 8 W is due to queue sizes below 16, where PCAP cannot achieve perfect capping (see Figure 7a). At higher utilization levels, large queues build up which allows PCAP to restore power to below (or close to) the target as shown for 9 W.

Discussion In summary, PCAP attains perfect power capping for high concurrency levels, whereas reductions in power usage are possible for lower levels. The effective concurrency for perfect capping tends to increase as the desired power cap decreases. We find that PCAP becomes effective starting from 32 threads. We also find that PCAP is ineffective below 4. We believe this should not be an immediate concern to the applicability of PCAP, since real systems are usually multi-threaded for performance reasons. As such, chances that little concurrency appears in practice are little. That said, throttling can be used in such cases.

8 Putting it all together

This section evaluates PCAP's performance under different workloads. Then, we evaluate it when capping power at the system level for an array of HDDs.

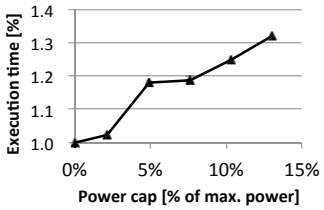

Figure 8: The increase in execution time for a 500 MB batch job for different power caps

8.1 PCAP under batch workloads

This section studies the impact of power capping by PCAP on the performance. We assume a batch job that reads data chunks from random locations on a single HDD. The HDD always reads at its maximum throughput and reads a total of 500 MB. We investigate the total execution time of the job when capping the HDD power at different levels. Since the job is batch, where throughput is important, we run PCAP in the throughput mode. And we vary the power cap in the range [8-9] W with a step of 0.25 W. We study the tail-latency mode later.

Figure 8 plots the execution time of the batch job versus the power cap. We normalize the figure to the case without power capping, where the average power is 9.2 W and the total execution time is 10.5 minutes. The figure shows that the execution time increases by 33% when capping at 13% (8 W), which is the maximum attainable cap. Also, the figure highlights a nonlinear relationship between the power cap and performance. For example, the increase in response time between power caps 2% and 5% is larger than that between 5% and 8% (16% vs. 2%). We confirmed this relationship by repeating the experiments and also examining the effective throughput of the application. We found that the throughput is 195 IOPS at 2% and drops to 167 IOPS and 165 IOPS at 5% and 8%, respectively. We study for bursty workloads next.

8.2 PCAP under bursty workloads

This section studies PCAP under workloads that vary in intensity and exhibit trough as well as bursty periods. This experiment seeks primarily to show that the increase in a job's execution time due to power capping (as shown in the previous section), can be absorbed in the subsequent trough periods. As such, long execution times do not necessarily always manifest. Still, power is capped.

We searched for real traces to replay but were challenged with two issues. The first issues was scaling the address space to reflect the growth in disk capacity. The second issue was scaling the arrival times. We obtained Microsoft traces [37] and implemented a replay tool. Although these traces confirm the randomness in the I/O

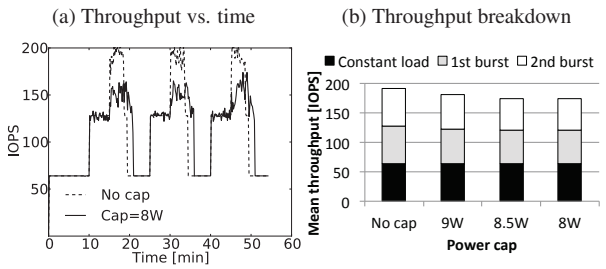

Figure 9: (a) PCAP performance under bursty workloads. The increase in execution time is absorbed, thanks to the trough periods. (b) Throughput is reduced due to capping and workload components share the cut.

pattern, we found that the address space is in the range of 256 GB (vs. our 4 TB HDDs). More importantly, we found that the traces were taken from a mostly-idle system, so that high power is not an issue but energy efficiency (see Section 2.1) for which indeed some techniques were proposed such as write offloading [37]. For example, we needed to scale traces by a factor up to 100 in order to see I/O activity, which in turn was not realistic because the inherent burstiness of the trace disappeared. We resorted to emulating a real workload.

Our workload is 55-minute long and consists of three parts. The first part is a constant part which continuously pushes load to the underlying HDD at a rate of 64 IOPS. The second and third parts are bursts which read 40 MB and 20 MB worth of random data, respectively. They are repeated 3 times throughout the workload separated by trough periods, each is 10-minute long. The third burst always starts 5 minutes after the second one. We repeated the experiment 4 times for different power caps: no cap, 9 W, 8.5 W, and 8 W. PCAP runs in the throughput mode.

Figure 9a shows the throughput in IOPS for the entire experiment. We show for the two extreme power settings: no cap and 8.0 W to keep the figure readable. Two observations can be made. First, the throughput during bursts decreases for power-capped scenarios, whereas it remains unaffected otherwise since the actual power is below the cap. The reduced throughput results in longer times to finish the burst jobs, which are perfectly absorbed in the subsequent trough periods. Secondly, both capping scenarios finish at the exact time of 55 minute. Note that in cases were no trough periods exist, longer execution times cannot be hidden and the discussion reduces to that of the previous section.

Figure 9b shows the split of the HDD's throughput across the three workload components. We show the split for the four power-capping settings. The throughput drops from 190 IOPS to 170 IOPS. The two bursty parts share the cut in throughput, 11% and 17%, respectively.

In summary, power capping impacts the performance of the HDD. In real scenarios, where the total execution time matters, a system like PCAP can incorporate performance as a target to optimize for while performing power capping. The resultant increase in response time manifests in high intensity workloads such as batch jobs, whereas it decreases in varying workloads.

8.3 PCAP/S: System-level power capping

This section demonstrates the application of power capping on larger scales with multiple HDDs. We present PCAP/S, PCAP for a system of HDDs.

PCAP/S builds on PCAP. It borrows the main control loop of PCAP and works with multiple HDDs. PCAP/S elects HDDs for power capping in a performance-aware way. To mimic real systems, we assume that HDDs are split into tiers, which represent different service level objectives (SLOs) as in service-oriented datacenters [35, 1] for instance. PCAP/S' power capping policy splits into two parts: (1) JBOD-level and (2) HDD-level. We chose for a priority-based strategy for the JBOD-level policy. It elects HDDs from the lower-priority tiers first when power must be reduced, whereas higher-priority tiers are elected first when more power is available. HDDs within the same tier are treated equally with regards to power increase or decrease. This policy strives to reduce performance penalty for higher-priority tiers. The HDD-level part, on the other hand, is exactly that of Section 6, which resizes queues to attain power capping per HDD.

We studied power capping for the array of HDDs for our JBOD from Section 3.1. We split the 16 HDDs in each JBOD into three different tiers. The first tier enjoys the best performance. Both Tier 1 and Tier 2 contain 5 HDDs each, whereas Tier 3 has 6. The JBOD itself consumes 80 W of power when unloaded (i.e., contains no HDDs). We set PCAP/S' period, $T = 10$ s and error margin, $\varepsilon = 1$ W.

We applied random workloads generated by FIO to the three tiers over a 55-minute period of time. Different power caps were used: (200, 190, 185, 170, 195, 150, 150, 180, 205 W.) PCAP/S was run in the latency mode. Figure 10a shows the total power differentiated into three tiers. Power capping starts at minute 10 with a power cap of 200 W (excl. the static power). PCAP/S reduces the power below the cap by reducing the consumption of Tier 3, the lowest-priority tier. At minute 20, however, the power cap is set at 185 W, which is larger than the maximum saving attainable by Tier 3. Therefore, Tier 2 takes a cut in power here too, but at a lesser degree than Tier 3. At minute 25, the power cap is set at 170 W, and Tiers 1–3 contribute to the reduction. When power is raised later at minute 30, Tier 1 regains its maximum power budget, whereas Tier 3 still observes a cut in power. At minute 35, a relatively very low power cap of 150 W is set, which is beyond the capping capability

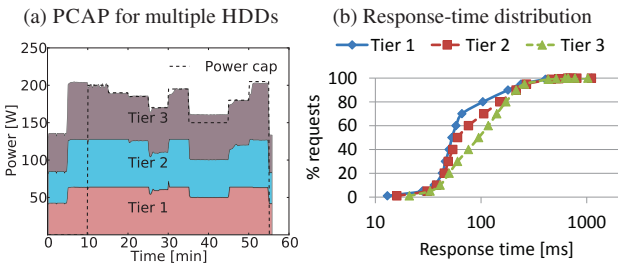

Figure 10: Using PCAP/S to (a) cap the power usage of our JBOD for (b) three tiers of performance

of PCAP/S. Here, PCAP/S does its best by maximizing the reduction on all the tiers, while being off by 10 W. As such, it caps power by up to 22% in this particular case.

Figure 10b shows the distribution of the service time for the three workloads applied on the three tiers, respectively. It shows that 80% of the requests observe latencies less than 104, 151, and 172 milliseconds on the three tiers, respectively. Also, 80%, 70% and 55% observe latencies under 100 ms, respectively. Thanks to the priority-based JBOD-level policy, higher-priority tiers suffer less performance penalties.

8.4 Discussion

PCAP works under mixed workloads as well as under random workloads observing the fact that mixed workloads result in a random pattern on the disk. The capping percentage will be affected as detailed in the experiments in Section 7.2. As for pure sequential workloads throttling can be used. Extending PCAP to detect sequential patterns and use our `dm-throttle` from Section 4 should be straightforward.

Queue resizing by PCAP is based on observations inherent to the fundamental architecture of the HDD (Section 5.1) as opposed to specifics of an HDD model or brand. Therefore, we expect that no per-device calibration is needed but perhaps per family of devices; regular versus Helium HDDs. Figure 1 confirms the similarity of the dynamic power range for contemporary HDDs from different vendors.

9 Related Work

Numerous studies have addressed the power and energy efficiency of IT systems. Some studies focus on mobile systems [32, 33, 46], while others focus on datacenters. The latter studies look into energy efficiency as well as power controllability and accountability. Energy-efficiency studies closely examine power management approaches for processors [35, 39, 25], memory [23, 25] and the disk drive. Approaches for the disk drive include power cycling [20, 24, 45, 48, 15], dynamic RPM [27, 15], buffering and scheduling [18, 31], and the acoustic seek mode [16]. Other studies addressed modeling the energy consumption of the disk drive for holistic system efficiency [28, 9, 46, 47]. Newer technologies were also investigated. Researchers looked into newer technologies, such as SSDs to reduce data movement costs using their energy-efficient computation [42].

Recently, power has received increased attention in an attempt to reduce running and infrastructure costs [36, 26, 38, 34, 30]. Some authors investigated power accounting on a per-virtual machine [30, 38, 41]. Other authors proposed techniques for power capping for the processor [34, 33] and the main memory [21]. As for the disk drive, the throughput-throttling technique [40] and the acoustic seek mode [16] were proposed. While throttling works under sequential workloads, it incurs large performance penalties for random workloads. Likewise, acoustic seeks result in slow seeks, which impacts performance too.

This work complements the previous work and propose queue resizing to cap the disk drive's power consumption under random and mixed workloads. We investigated the relationship between the queue size and the power usage of the disk drive. We showed that queues can be resized to cap power yet in a performance-aware manner. We designed PCAP based on key observations of the queue–power relationship. PCAP is capable of capping for single- and multi-HDD systems. We made the case for PCAP's superiority over throttling.

10 Summary

We presented a technique to cap the power usage of 3.5-inch disk drives. The technique is based on queue resizing. We presented PCAP, an agile system to cap power for the disk drive. We evaluated PCAP performance on a system of 16 disks. We showed that PCAP outperforms throttling. In our experiments, 60% of the requests exhibit response times below 100 ms and an increase of 32% in throughput is attainable with PCAP. We also showed that PCAP caps power for tiered storage systems and offers performance-differentiation on larger scales.

Acknowledgments

The authors wish to thank Adam Manzanares for his thoughtful comments on early drafts of this paper. We also thank Jim Donaldson for helping with setting up the measurement apparatus. Our shepherd, Ken Salem, and the anonymous FAST reviewers helped improving the clarity of the manuscript with their detailed comments.

References

[1] Amazon S3. http://aws.amazon.com/s3/.

[2] HGST 3.5-inch Enterprise Hard Drive. http://www.hgst.com/hard-drives/enterprise-hard-drives/enterprise-sata-drives/ultrastar-7k4000.

[3] Hgst active archive system. http://www.hgst.com/products/systems/hgst-active-archive-system.

[4] mongoDB. http://www.mongodb.org/.

[5] Native Command Queuing. https://www.sata-io.org/native-command-queuing.

[6] Seagate Enterprise Capacity 3.5 HDD. http://www.seagate.com/internal-hard-drives/enterprise-hard-drives/hdd/enterprise-capacity-3-5-hdd/.

[7] Wattsup .Net power meter. https://www.wattsupmeters.com/secure/products.php?pn=0&wai=0&more=2.

[8] WD Enterprise Re 3.5 HDD. http://www.wdc.com/en/products/products.aspx?id=580.

[9] ALLALOUF, M., ARBITMAN, Y., FACTOR, M., KAT, R. I., METH, K., AND NAOR, D. Storage modeling for power estimation. In *Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference* (New York, NY, USA, 2009), SYSTOR'09, ACM, pp. 3:1–3:10.

[10] AXBOE, J. FIO benchmark. http://freecode.com/projects/fio.

[11] BARROSO, L. A. The price of performance. *ACM Queue 3*, 7 (Sept. 2005), 48–53.

[12] BARROSO, L. A., CLIDARAS, J., AND HÖLZLE, U. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, Second edition*, vol. 8. 2013.

[13] BARROSO, L. A., DEAN, J., AND HÖLZLE, U. Web Search for a Planet: The Google Cluster Architecture. *IEEE Micro 23*, 2 (Mar. 2003), 22–28.

[14] BOVET, D., AND CESATI, M. *Understanding The Linux Kernel*. Oreilly & Associates Inc, 2005.

[15] CARRERA, E. V., PINHEIRO, E., AND BIANCHINI, R. Conserving disk energy in network servers. In *Proceedings of the 17th Annual International Conference on Supercomputing* (New York, NY, USA, 2003), ICS'03, ACM, pp. 86–97.

[16] CHEN, D., GOLDBERG, G., KAHN, R., KAT, R. I., AND METH, K. Leveraging disk drive acoustic modes for power management. In *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies* (Washington, DC, USA, 2010), MSST'10, IEEE Computer Society, pp. 1–9.

[17] CHEN, Y., ALSPAUGH, S., BORTHAKUR, D., AND KATZ, R. Energy efficiency for large-scale mapreduce workloads with significant interactive analysis. In *Proceedings of the 7th ACM european conference on Computer Systems* (New York, NY, USA, 2012), EuroSys '12, ACM, pp. 43–56.

[18] CHOI, J., WON, Y., AND NAM, S. W. Power conscious disk scheduling for multimedia data retrieval. In *Proceedings of the 2nd International Conference on Advances in Information Systems* (2002), ADVIS'02, pp. 336–345.

[19] COOPER, B. F., SILBERSTEIN, A., TAM, E., RAMAKRISHNAN, R., AND SEARS, R. Benchmarking Cloud Serving Systems with YCSB. In *Proceedings of the 1st ACM Symposium on Cloud Computing* (New York, NY, USA, 2010), SoCC'10, ACM, pp. 143–154.

[20] CRAVEN, M., AND AMER, A. Predictive Reduction of Power and Latency (PuRPLe). In *Proceedings of the 22nd IEEE / 13th NASA Goddard Conference on Mass Storage Systems and Technologies* (2005), MSST'05, pp. 237–244.

[21] DAVID, H., GORBATOV, E., HANEBUTTE, U. R., KHANNA, R., AND LE, C. RAPL: Memory power estimation and capping. In *Proceedings of 2010 ACM/IEEE International Symposium on Low-Power Electronics and Design* (Aug 2010), ISLPED'10, pp. 189–194.

[22] DEAN, J., AND BARROSO, L. A. The Tail at Scale. *Communications of the ACM 56* (2013), 74–80.

[23] DENG, Q., MEISNER, D., RAMOS, L., WENISCH, T. F., AND BIANCHINI, R. Memscale: Active low-power modes for main memory. In *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems* (New York, NY, USA, 2011), ASPLOS XVI, ACM, pp. 225–238.

[24] DOUGLIS, F., KRISHNAN, P., AND MARSH, B. Thwarting the power-hungry disk. In *Proceedings of the USENIX Winter 1994 Technical Conference* (Berkeley, CA, USA, 1994), WTEC'94, USENIX Association, pp. 293–306.

[25] FAN, X., ELLIS, C. S., AND LEBECK, A. R. The synergy between power-aware memory systems and processor voltage scaling. In *Proceedings of the Third International Conference on Power - Aware Computer Systems* (Berlin, Heidelberg, 2004), PACS'03, Springer-Verlag, pp. 164–179.

[26] FAN, X., WEBER, W.-D., AND BARROSO, L. A. Power provisioning for a warehouse-sized computer. In *Proceedings of the 34th Annual International Symposium on Computer Architecture* (New York, NY, USA, 2007), ISCA'07, ACM, pp. 13–23.

[27] GURUMURTHI, S., SIVASUBRAMANIAM, A., KANDEMIR, M., AND FRANKE, H. DRPM: dynamic speed control for power management in server class disks. In *Proceedings of the 30th Annual International Symposium on Computer Architecture* (New York, NY, USA, 2003), ISCA'03, ACM, pp. 169–181.

[28] HYLICK, A., SOHAN, R., RICE, A., AND JONES, B. An analysis of hard drive energy consumption. In *Proceedings of the 16th Annual Meeting of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS'08* (2008), IEEE Computer Society, pp. 103–112.

[29] IV, W. T., SEADER, J., AND BRILL, K. Tier classifications define site infrastructure performance. *The Uptime Institute, White Paper* (2006).

[30] KANSAL, A., ZHAO, F., LIU, J., KOTHARI, N., AND BHATTACHARYA, A. A. Virtual machine power metering and provisioning. In *Proceedings of the 1st ACM Symposium on Cloud Computing* (New York, NY, USA, 2010), SoCC'10, ACM, pp. 39–50.

[31] KHATIB, M. G., VAN DER ZWAAG, B. J., HARTEL, P. H., AND SMIT, G. J. M. Interposing Flash between Disk and DRAM to Save Energy for Streaming Workloads. In *IEEE/ACM/IFIP Workshop on Embedded Systems for Real-Time Multimedia, 2007. ESTIMedia 2007* (Oct 2007), pp. 7–12.

[32] KIM, H., AGRAWAL, N., AND UNGUREANU, C. Revisiting storage for smartphones. In *Proceedings of the 10th USENIX Conference on File and Storage Technologies* (Berkeley, CA, USA, 2012), FAST'12, USENIX Association, pp. 209–222.

[33] LI, J., BADAM, A., CHANDRA, R., SWANSON, S., WORTHINGTON, B., AND ZHANG, Q. On the Energy Overhead of Mobile Storage Systems. In *Proceedings of the 12th USENIX Conference on File and Storage Technologies* (Santa Clara, CA, 2014), FAST'14, USENIX Association, pp. 105–118.

[34] LIM, H., KANSAL, A., AND LIU, J. Power budgeting for virtualized data centers. In *Proceedings of the 2011 USENIX Conference on Annual Technical Conference* (Berkeley, CA, USA, 2011), ATC'11, USENIX Association, pp. 59–72.

[35] LO, D., CHENG, L., GOVINDARAJU, R., BARROSO, L. A., AND KOZYRAKIS, C. Towards Energy Proportionality for Large-scale Latency-critical Workloads. In *Proceeding of the 41st Annual International Symposium on Computer Architecuture* (Piscataway, NJ, USA, 2014), ISCA'14, IEEE Press, pp. 301–312.

[36] MEISNER, D., SADLER, C. M., BARROSO, L. A., WEBER, W.-D., AND WENISCH, T. F. Power management of online data-intensive services. In *Proceedings of the 38th Annual International Symposium on Computer Architecture* (New York, NY, USA, 2011), ISCA'11, ACM, pp. 319–330.

[37] NARAYANAN, D., DONNELLY, A., AND ROWSTRON, A. Write off-loading: Practical power management for enterprise storage. *Trans. Storage 4*, 3 (Nov. 2008), 10:1–10:23.

[38] RAGHAVENDRA, R., RANGANATHAN, P., TALWAR, V., WANG, Z., AND ZHU, X. No "power" struggles: Coordinated multi-level power management for the data center. In *Proceedings of the 13th International Conference on Architectural Support for Programming Languages and Operating Systems* (New York, NY, USA, 2008), ASPLOS XIII, ACM, pp. 48–59.

[39] RAJAMANI, K., LEFURGY, C., GHIASI, S., RUBIO, J., HANSON, H., AND KELLER, T. Power management solutions for computer systems and datacenters. In *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design* (Aug 2008), ISLPED'08, pp. 135–136.

[40] STOESS, J., LANG, C., AND BELLOSA, F. Energy management for hypervisor-based virtual machines. In *Proceedings of the USENIX Annual Technical Conference* (Berkeley, CA, USA, 2007), ATC'07, USENIX Association, pp. 1:1–1:14.

[41] THERESKA, E., DONNELLY, A., AND NARAYANAN, D. Sierra: practical power-proportionality for data center storage. In *Proceedings of the sixth conference on Computer systems* (New York, NY, USA, 2011), EuroSys'11, ACM, pp. 169–182.

[42] TIWARI, D., VAZHKUDAI, S. S., KIM, Y., MA, X., BOBOILA, S., AND DESNOYERS, P. J. Reducing Data Movement Costs Using Energy-Efficient, Active Computation on SSD. In *Presented as part of the 2012 Workshop on Power-Aware Computing and Systems* (Berkeley, CA, 2012), USENIX.

[43] VERMA, A., KOLLER, R., USECHE, L., AND RANGASWAMI, R. SRCMap: energy proportional storage using dynamic consolidation. In *Proceedings of the 8th USENIX conference on File and storage technologies* (Berkeley, CA, USA, 2010), FAST'10, USENIX Association, pp. 267–280.

[44] WEDDLE, C., OLDHAM, M., QIAN, J., WANG, A.-I. A., REIHER, P., AND KUENNING, G. PARAID: A gear-shifting power-aware RAID. *Trans. Storage 3*, 3 (Oct 2007).

[45] XU, L., CIPAR, J., KREVAT, E., TUMANOV, A., GUPTA, N., KOZUCH, M. A., AND GANGER, G. R. SpringFS: Bridging Agility and Performance in Elastic Distributed Storage. In *Proceedings of the 12th USENIX conference on File and storage technologies* (Berkeley, CA, USA, 2014), FAST'14, USENIX Association, pp. 243–256.

[46] ZEDLEWSKI, J., SOBTI, S., GARG, N., ZHENG, F., KRISHNAMURTHY, A., AND WANG, R. Modeling hard-disk power consumption. In *Proceedings of the 2nd USENIX Conference on File and Storage Technologies* (Berkeley, CA, USA, 2003), FAST'03, USENIX Association, pp. 217–230.

[47] ZHANG, Y., GURUMURTHI, S., AND STAN, M. R. SODA: Sensitivity Based Optimization of Disk Architecture. In *Proceedings of the 44th Annual Design Automation Conference* (New York, NY, USA, 2007), DAC'07, ACM, pp. 865–870.

[48] ZHU, Q., CHEN, Z., TAN, L., ZHOU, Y., KEETON, K., AND WILKES, J. Hibernator: helping disk arrays sleep through the winter. In *Proceedings of the twentieth ACM symposium on Operating systems principles* (New York, NY, USA, 2005), SOSP'05, ACM, pp. 177–190.