MINIMIZING POWER CONSUMPTION IN VIDEO SERVERS BY THE COMBINED USE OF SOLID-STATE DISKS AND MULTI-SPEED DISKS

Seminar Report

Submitted in partial fulfillment of the requirements for the award of degree of

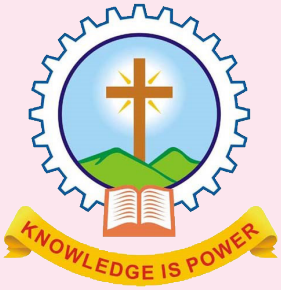
BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE AND ENGINEERING

of

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

Submitted By

PANCHAMI A MOHAN



Department of Computer Science & Engineering
Mar Athanasius College of Engineering Kothamangalam

MINIMIZING POWER CONSUMPTION IN VIDEO SERVERS BY THE COMBINED USE OF SOLID-STATE DISKS AND MULTI-SPEED DISKS

Seminar Report

Submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY

In

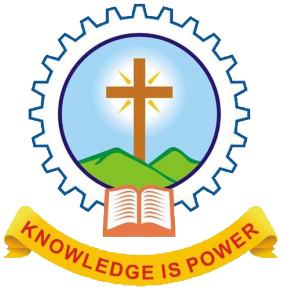
COMPUTER SCIENCE AND ENGINEERING

of

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

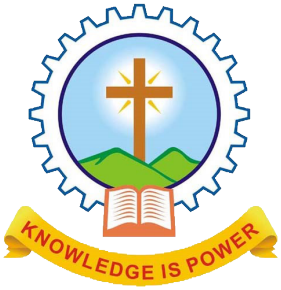
Submitted By

PANCHAMI A MOHAN



Department of Computer Science & Engineering
Mar Athanasius College of Engineering Kothamangalam

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MAR ATHANASIUS COLLEGE OF ENGINEERING
KOTHAMANGALAM



CERTIFICATE

This is to certify that the report entitled **Minimizing Power Consumption in Video Servers by the Combined Use of Solid-State Disks and Multi-Speed Disks** *submitted by* **Ms. PANCHAMI A MOHAN**, *Reg. No.* **MAC15CS045** *towards partial fulfilment of the requirement for the award of Degree of Bachelor of Technology in Computer science and Engineering from APJ Abdul Kalam Technological University for the year December 2018 is a bonafide record of the work carried out by her under our supervision and guidance.*

.....	.....	.....
Prof. Joby George	**Prof. Neethu Subash**	**Dr.Surekha Mariam Varghese**
Faculty Guide	***Faculty Guide***	***Head of the Department***

Date:

Dept. Seal

ACKNOWLEDGEMENT

First and foremost, I sincerely thank the God Almighty for his grace for the successful and timely completion of the seminar.

I express my sincere gratitude and thanks to Dr. Solly George, Principal and Dr. Surekha Mariam Varghese, Head Of the Department for providing the necessary facilities and their encouragement and support.

I owe special thanks to the staff-in-charge Prof. Joby George, Prof. Neetha Subash and Prof. Joby Anu Mathew for their corrections, suggestions and sincere efforts to co-ordinate the seminar under a tight schedule.

I express my sincere thanks to staff members in the Department of Computer Science and Engineering who have taken sincere efforts in helping me to conduct this seminar.

Finally, I would like to acknowledge the heartfelt efforts, comments, criticisms, co-operation and tremendous support given to me by my dear friends during the preparation of the seminar and also during the presentation without whose support this work would have been all the more difficult to accomplish.

ABSTRACT

Disks use a considerable portion of the total energy consumed by a video-server. In order to reduce the high power consumption, solid-state disks (SSDs) and multi-speed disks are introduced. Using multi-disks running at different speeds can greatly reduce power consumption even under server workloads. SSD is used as a cache to improve the throughput of a video server and to minimize power consumption. Effective SSD bandwidth management is essential for satisfying high request rates for popular videos stored on SSD. An SSD bandwidth allocation algorithm is proposed to minimize energy consumption and allows disks to run at lower speeds. This result in a jitter-free disk speed transition. It is validated that the proposed bandwidth allocation algorithm achieves appreciable power savings under various workloads and limits the number of disk speed changes.

Contents

List of Figures

List of Tables

List of Abbreviations

SSD	Solid State Disk
HTTP	HyperText Transfer Protocol
RAID	Redundant Array of Independent Disk
LKP	Linear Knapsack Problem
MMKP	Multiple-choice Knapsack Problem
BAP	Bandwidth Allocation Problem
LS	Lowest bit-rate first selection
US	Uniform selection

Introduction

Recent advances in network and system technologies have made it feasible to provide video-on-demand (VoD) services for a range of applications including digital libraries, education-on-demand, distance learning, and user-created content. Naturally, this increases the amount of Internet video traffic, which then requires a large server structure. For example, it has been reported that nearly five billion videos are watched everyday on YouTube.

This growing demand for video data increases the energy consumed by servers, which represents a major problem. In particular, because of their high bandwidth requirements, servers are typically built as a redundant array of independent disk (RAID), which may comprise thousands of disks, thereby making it one of the biggest energy consumers in a server. Recent studies have shown that the energy consumed by storage systems may constitute 40% of that of the entire data center. Considerable energy consumption also represents a serious economic concern for service providers. For instance, a data center with a provisioned 10-MW peak power capacity costs between 100 and 200 million dollars per year .

A major consequence of this high power consumption is heat, which negatively affects system reliability. For example, it has been shown that running at 15^oC above ambient temperature can double the failure rate of disk drives. However, cooling systems for high heat densities are prohibitively expensive and the cooling system itself adds considerably to the power requirement. For example, some studies have shown that the cooling system is the largest power consumer in a data center, where it accounts for 50% of the total power used. High energy consumption also hinders the expansion of servers and has a negative effect on the environment.

Reducing the speed at which a disk spins decreases its power consumption. Thus, Gurumurthi and Gurumurthi et al. proposed using disks that can run at several different speeds. Using multiple disks running at different speeds can greatly reduce power consumption, even under video server workloads, as the disks spin slowly in response to fewer video requests. By contrast, a singlespeed disk always spins at full speed. Multi-speed disks are commercially available. These include drives produced by Western Digital that use an IntelliPower technique to adjust the rotations per minute (RPM) and transfer rate to reduce the disk power consumption. They have also been used to build more energy-efficient storage systems (e.g., Nexsan storage servers. These developments have motivated further research into effective power reduction techniques based on multi-speed disks.

Flash-based solid-state disks (SSDs) have many technical merits such as non-volatility, low power consumption, shock resistance, and excellent random read performance, and thus are now being used for enterprise storage systems such as video servers. The main reasons for using SSDs in video servers include: 1) low latency when reading data, which is crucial in video servers; 2) good throughput for large random read operations, which is important for supporting large numbers of simultaneous video streams; and 3) good performance for sequential write operation, which is a necessary attribute of a video server.

The high cost of SSDs prohibits their use for storage in video servers because of the amount of data in videos (e.g., storing 10,000 2 h movies at 9 Mb/s requires 78 TB). The cost of flash memory has been declining, but it is still an order of magnitude more expensive than disk storage. However, the pattern of access to a set of videos tends to be highly skewed because most requests are for only a few popular videos. Thus, we can use an SSD as a cache to improve the throughput of a video server while also reducing the disk power consumption rate. However, this also means that making effective use of the limited SSD bandwidth is essential due to the high request rates for popular video files on the SSD.

Several studies have attempted to use SSDs for video servers. For example, a file-level caching scheme was recently introduced to increase the number of simultaneous clients, and a dynamic data replication scheme was developed for allowing popular videos to be replicated on SSD, so as to improve the SSD caching hit ratio. However, to the best of our knowledge, the present study is the first to consider SSD bandwidth management issues in order to minimize power consumption for a video server that uses SSDs and multi-speed disks.

We present an SSD cache management scheme for video servers that use multi-speed disks. First, we examine how the allocation of SSD storage affects disk bandwidth. We then examine how the power consumption varies with the number of requests served by the SSD and propose an algorithm that determines the requests to be served from the SSD in order to minimize the disk energy consumption subject to the SSD bandwidth. This algorithm also considers disk reliability issues by limiting the number of speed changes and by providing speed transitions without jitter.

Related works

2.1 Disk power management for servers

Techniques for managing disk power consumption in servers have been studied widely, mainly by extending the periods when the disk can be in a low-power mode. Several schemes redistribute the workload to allow more disks to remain in low-power modes. For example, Pinheiro and Bianchini[1] proposed a data concentration technique in which the workloads are dynamically migrated so the load becomes skewed toward only a few disks in order to allow other disks to enter low-power modes. Khatib and Bandic proposed a power capping scheme that uses different disk scheduling algorithms to limit disk power consumption.

Redundancy is essential for ensuring fault tolerance in all practical storage systems, and this approach has been exploited by several energy conservation schemes. Pinheiro et al. presented a request-redirection scheme to leverage the redundancy in storage systems. Weddle et al. proposed a new RAID architecture that employs skewed striping techniques to maximize the number of disks in the low-power mode.

To reduce the amount of energy used when server work- loads are intense, Gurumurthi and Gurumurthi et al.[2] proposed a speed-changing scheme called dynamic RPM and suggested that single-speed disks are not suitable for reducing the energy used when processing server workloads. Several schemes have been presented for servers based on this method. Zhu et al. proposed a flexible disk placement architecture called Hibernator to reduce disk speeds as often as possible. Xie described a data placement technique in which the loads are distributed such that some disks can run at low speeds.

All of these techniques attempt to resolve disk power issues using various methods, but they cannot guarantee real- time data retrieval, thus making them unsuitable for video servers. Lee et al. presented a prefetching scheme in which the amount of data prefetched for each video stream is dynamically adjusted to minimize the power consumption by a disk array with heterogeneous disks. Song sug- gested using multi-speed disks for video servers by guaran- teeing continuous data retrieval when disk speeds change. Kim and Song[3] proposed a technique that adjusts the retrieval period to allow more disks to run at low speeds. Yuan et al. presented energy management techniques to satisfy delay constraints for large-scale video-sharing services. However, none of these methods use SSD.

2.2 Use of Solid-State Disk for servers

Many studies have used an SSD cache to improve the performance of disk-based storage systems. Li et al.[4] proposed a caching architecture with adjustable deduplication, compression, and replacement granularities in order to minimize the response time. Arteaga et al. developed schemes for allocating flash memory across virtual machines to balance the cache loads in cloud computing systems. Meng et al. developed schemes for partitioning flash memory to optimize the allocation of flash memory to virtual machines.

To utilize the limited SSD space effectively, various meth- ods of storing popular data on the SSD have been studied. Kim and Kim suggested a method of storing the prefix parts of popular videos onto SSD to improve overall data bandwidth. Lin et al. introduced a scheme called hot random offloading (HRO) that stores frequently accessed random data on an SSD to handle as many random I/O requests as possible from the SSD. Yin et al. presented an SSD cache architecture that stores popular data on the SSD in order to allow disks to spin down when the workloads are light, thereby reducing the amount of energy consumed by the disk. He et al. proposed a new SSD cache placement scheme in which performance-critical data are cached on an SSD to fully utilize SSD-based file servers with the aim of improving I/O performance of the parallel file systems.

SSDs can also be used to reduce the power consumed by disk arrays. Kgil et al. studied various types of cache architecture in order to reduce power consumption by the main memory and disk. Useche et al. presented several schemes for an energy-aware SSD cache. These included data popularity prediction, I/O indirection, and reconfiguration schemes. Hui et al.[5] introduced a hybrid storage system that uses an SSD as a cache to allow under-utilized disks to be spun down.

None of the methods previously mentioned consider multimedia workloads. To address this, Ryu et al. analyzed the advantages and disadvantages of using SSDs in video servers, where it was shown that low-end SSDs can allow essentially constant throughput. They also extended this approach to the use of SSDs for HTTP adaptive streaming. Manjunath and Xie considered an architecture for an SSD-based server in which a video file can be dynamically replicated from disk to SSD in order to reduce the disk load. Lee and Song developed a video cache management scheme that considers the popularity of video segments in order to minimize disk bandwidth consumption. Chen et al.[6] developed an error correction technique to use low-cost flash memory for video servers by considering video coding and flash memory characteristics. In all of these previous studies, however, disk power issues were not considered.

Solid-state disks

A typical SSD uses what is called NAND-based flash memory. This is a non-volatile type of memory. What does non-volatile mean you ask? The simple answer is that you can turn off the disk and it wont forget what was stored on it. This is of course an essential characteristic of any type of permanent memory. During the early days of SSD, rumors floated around saying stored data would wear off and be lost after only a few years. Regardless, that rumor is certainly not true with todays technology, as you can read and write to an SSD all day long and the data storage integrity will be maintained for well over 200 years.

An SSD does not have a mechanical arm to read and write data, it instead relies on an embedded processor (or brain) called a controller to perform a bunch of operations related to reading and writing data. The controller is a very important factor in determining the speed of the SSD. Decisions it makes related to how to store, retrieve, cache and clean up data can determine the overall speed of the drive. We wont get into the nitty-gritty details for the various tasks it performs such as error correction, read and write caching, encryption, and garbage collection to name a few. Yet, suffice to say, good controller technology is often what separates an excellent SSD from a good one. An example of a fast controller today is the SandForce SATA 3.0 (6GB/s) SSD controller that supports burst speeds up to 550MB/s read and write speeds. The next gen SandForce 3700 family of controllers was announced in late 2013, and is quoted to reach a blistering 1,800MB/s read/write sequential speeds as well as 150K/80K random IOPS.

Solid-State Disk architecture

A solid-state disk (SSD) is a solid-state storage device that uses integrated circuit assemblies as memory to store data persistently. It is also sometimes called solid-state disk, although SSDs do not have physical disks. SSDs may use traditional hard disk drive (HDD) form-factors and protocols such as SATA and SAS, greatly simplifying usage of SSDs in computers. Following the initial acceptance of SSDs with HDD interfaces, new form factors such as the M.2 form factor, and new I/O protocols such as NVM Express have been developed to address specific requirements of the flash memory technology used in SSDs.

As shown in Fig.2.1, the key components of an SSD are the controller and the memory to store the data. The primary memory component in an SSD was traditionally DRAM volatile memory, but it is now more commonly NAND flash non-volatile memory. Most SSD manufacturers use non-volatile NAND flash memory in the construction of their SSDs because of the lower cost compared with DRAM and the ability to retain the data without a constant power supply, ensuring data persistence through sudden power outages. Flash memory SSDs

are slower than DRAM solutions, and some early designs were even slower than HDDs after continued use.

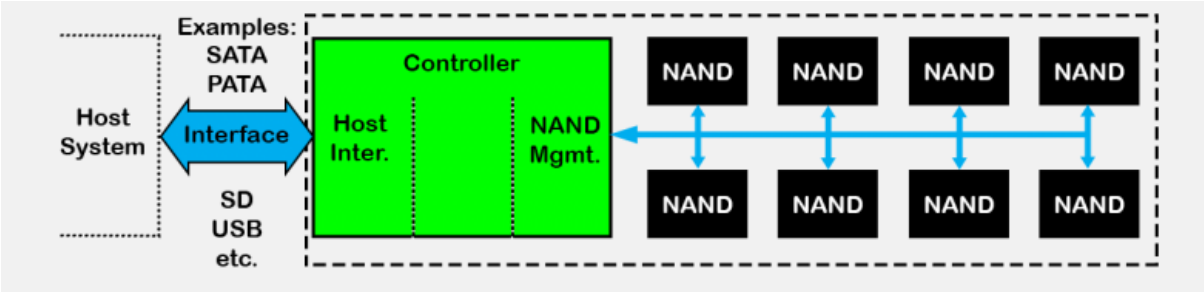


Fig. 2.1: Basic SSD architecture.

The reason for the controller function is to manage the NAND components and create a standard interface which interfaces well to host systems. There are many popular interfaces today such as Serial ATA (SATA), SD, MMC, USB, PCIe as well as Parallel ATA (PATA, aka IDE). All of these SSD interfaces have a common controller architecture design in which a controller resides between the NAND memory and the host system. In future articles we will look at the tasks a controller handles, but here we focus on the basic architecture of a generic Solid State Drive (SSD).

Proposed system

If I/O requests are served by the SSD cache, then disk bandwidth utilization at each speed level is effectively reduced, which also decreases the power consumed by the disk. However, effective SSD bandwidth management is essential to serve many popular requests from the cache. To address this problem, we propose an algorithm that determines requests served by the SSD based on the following steps:

1. The problem formulation step examines how the power consumed at each disk speed level depends on the number of requests served by the SSD.

2. The bandwidth allocation step determines the requests served by the SSD to minimize the overall power con- sumption while also limiting the number of disk speed changes.

Frequently starting and stopping disks is known to affect disk drive longevity. Therefore, many disk specifications provide their duty cycles[7], which are the maximum number of times that the drive can be spun down. The disk speed change also involves this spin-down operation, which means that minimizing the number of speed changes is essential. Therefore, we limit the number of simultaneous speed changes when the algorithm is executed.

3.1 System model

Consider a video server composed of an SSD and an array of multi-speed disks. When the server receives video requests, it first checks whether requested data is located on the SSD; otherwise, video data can be served from multi-speed disks.

In a video-on-demand environment[5], continuous delivery of video streams to the clients is guaranteed by sufficient reserved network and server resources. This leads to a hard limit on the number of streams that a video server can deliver. Multiple client requests for the same video can be served with a single disk I/O stream by sending (multicasting) the same data blocks to multiple clients (with the multicast facility, if present in the system). This is achieved by batching (grouping) requests for the same video that arrive within a short time. We explore the role of customerwaiting time and reneging behavior in selecting the video to be multicast. Table 3.1 summarizes the important symbols used in equations.

Table 3.1: Important symbols used

NOTATIONS	MEANING
N_s, N_v, N_d	Numbers of speeds, disks and videos repectively
θ	Skewness parameters in Zipf distribution
$P_s(l), P_a(l), P_i(l)$	Seek, active and idle power for speed level l, respectively
T_r, T_s	Round length and typical seek time respectively
T_t	Transition time between two different speed levels
Pt	Power required for speed transition
$r(l)$	Disk transfer rate at speed level l
S_i, b_i, P_i, L_i	Size, bit-rate, access probability and length of video i, respectively
D_i	Average disk bandwidth utilization for over all speed levels for a video i
G_i	Proportion of disk bandwidth utilization reduction for a video i over all videos
Ss_{sd}, B_{ssd}	Capacity and Bandwidth of videos repectively
$V(j)$	Video index for request j
A_k	Array of possible request combinations served by SSD for a disk k
$C_{k,m}$	Set of requests in the mth combination array A_k
N_k	Number of elemts in array A_k
$X_{k,m}$	Binary variable that determines whether request in the mth element of A_k are served by SSD
$D_{k,m}(l)$	Disk bandwidth utilization of disk k at speed level l when $X_{k,m} = 1$
$E_{k,m}^{seek}(l), E_{k,m}^{idle}(l), E_{k,m}^{active}(l)$	Energy consumption in seek, active and idle phases at speed level l during a round when $X_{k,m} = 1$
$B_{k,m}$	Amount of SSD bandwidth required for a set of requests in $Ck_{,m}$
$L_{k,m}$	Lowest speed level for disk k when $X_{k,m} = 1$
N_p	Number of rounds allocated for prefetching
$D_{k,m}^{cont}(l)$	Disk Bandwidth Utilization for disk k at speed level l reserved for refecting during the speed transition when $X_{k,m} = 1$
$L_{k,m}$	Lowest speed level for disk k where $\{l \mid D_{k,m}(l) + D_{k,m}^{cont}(l) \leq 1\}$
$ch_{k,m}$	Variable indicating whether speed changes occur when $X_{k,m} = 1$
N_c	Number of simultaneous speed changes allowed
U	Upper bound on speed changes
α	Every α th execution of the algorithm, where Nc is set to U; otherwise Nc=0

We organize the retrieval of data from a video server using round-based scheduling[6]. The time is divided into periods called rounds which are of an equal length of T_r , where each client is served once during each round. Video file i has a bit rate of b_i . When a server receives requests from clients, it allocates bandwidth to each request based on its bit rate. The amount of data that must be read for each video i during round T_r is $b_i T_r$ in order to maintain the playback rate. Therefore, to stream at 1.5 Mbps with a round length of 2 s, the server must read 3 Mbits of data during every round.

Since the data transfer rate of a single disk is significantly higher than the playback rate of a single video stream, each disk in a video server is able to provide the data for multiple streams. Disk bandwidth utilization [7]can be defined as the ratio of the total service time that a disk spends retrieving all the streams during a round relative to the round length, and it must be less than or equal to 1.

We consider a server that contains N_d multi-speed disks, in which each disk supports N s speed levels. Let r(l) be the data transfer rate for a disk running at speed level l, (l = 1, . . . , N_s). We use a typical seek-time model[7] in which a constant seek time T_s is required for one read of contiguous data. The disk head can start reading as soon as it reaches the destination track; it then reads the entire track rather than waiting to go to the location in the track where data is placed. Therefore, we assume that the rotational delay is 0. Reading the video file incurs a seek overhead of T_s and a reading time of $b_i T_r$/r(l), which increases the service time by $T_s + \frac{b_i T_r}{r(l)}$. Thus, serving video i increases the disk bandwidth utilization at speed l by $\frac{T_s + \frac{b_i T_r}{r(l)}}{T_r}$. Let D_i be the average disk bandwidth utilization over all speed levels for video i, which can be calculated as follows:

$$D_i = \frac{\sum_{l=1}^{N_s} T_s + \frac{b_i T_r}{r(l)}}{T_r N_s} \qquad \text{(Equ : 3.1)}$$

3.2 Solid-State Disk storage allocation

A small fraction of videos attract most of the user interest, whereas the vast majority of videos are of limited views. Given the huge amount of video content and the high variability of user attention, it is of almost importance for a number of tasks to understand the characteristics of online video popularity and further predict the popularity of individual videos.The arrival of client requests follows a Poisson distribution[5] and the access probability follows a

Zipf distribution. Therefore, the probability of requiring video i, p_i , is calculated as follows:

$$p_i = \frac{1}{\sum_{m=1}^{N_v} m^{\frac{1}{1-\theta}}} N_v \frac{1}{i^{i-\theta}} \qquad \text{(Equ : 3.2)}$$

where N_v is the number of videos and is a skewness parameter[6] . Let L_i be the length of video i in seconds. The proportion of requests for video i over all the videos can then be expressed as $p_i L_i$. Let G_i be a parameter that denotes the expected proportion of disk bandwidth utilization reduction, which can be expressed as:

$$G_i = p_i L_i D_i \qquad \text{(Equ : 3.3)}$$

Let S_i be the size of video i in MB. Let Y_i denote whether the first Y_i proportion of video i is cached, ($0 \leq$ Y i ≤ 1). Obviously, our aim is to cache video files in order to maximize the disk bandwidth reduction, $\sum_{i=1}^{N_v} Y_i G_i$. If Y_i = 1, then video i is entirely cached. By contrast, if Y_i = 0, video i is not cached at all. Let S_{ssd} be the size of SSD in MB, and the SSD storage requirement must not exceed S_{ssd} . Thus, $\sum_{i=1}^{N_v} Y_i S_i \leq S_{ssd}$. We can formulate the SSD allocation problem that determines Y_i as follows:

$$\text{Maximize} \sum_{i=1}^{N_v} Y_i G_i$$

$$\text{subject to} \sum_{i=1}^{N_v} Y_i S_i \leq S_{ssd}$$

This problem is a linear version of the 0/1 knapsack problem (LKP)[2]. The knapsack problem or rucksack problem is a problem in combinatorial optimization: Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. It derives its name from the problem faced by someone who is constrained by a fixed-size knapsack and must fill it with the most valuable items.

Given weights and values of n items, put these items in a knapsack of capacity W to get the maximum total value in the knapsack. In other words, given two integer arrays val[0....n-1] and wt[0..n-1] which represent values and weights associated with n items respectively. Also given an integer W which represents knapsack capacity, we find the maximum value subset of val[] such that sum of the weights of this subset is smaller than or equal to W. We cannot break an item, either pick the complete item, or dont pick it (0-1 property). In the LKP, each object

has a weight and profit, and the problem involves selecting a fractional pro- portion of each object such that the total profit is maximized while satisfying the capacity constraints. We can treat the storage allocation problem as the LKP by considering the SSD as a knapsack and by regarding each video as an object.

The optimal solution to the LKP can be found using a greedy approach. Thus, in the storage allocation prob- lem, each video is sorted in non-ascending order of the values of G_i/S_i , and the video files having higher values for G_i/S_i are cached first subject to the SSD storage limit.

3.2.1 Poisson distribution

The Poisson distribution[4] is the discrete probability distribution of the number of events occurring in a given time period, given the average number of times the event occurs over that time period.

The Poisson distribution is applicable only when several conditions hold.

- An event can occur any number of times during a time period.
- Events occur independently. In other words, if an event occurs, it does not affect the probability of another event occurring in the same time period.
- The rate of occurrence is constant; that is, the rate does not change based on time.
- The probability of an event occurring is proportional to the length of the time period. For example, it should be twice as likely for an event to occur in a 2 hour time period than it is for an event to occur in a 1 hour period.

Multi-speed disks

The platters in contemporary HDDs are spun at speeds varying from 4,200 rpm in energy-efficient portable devices, to 15,000 rpm for high-performance servers. The first HDDs spun at 1,200 rpm[6] and, for many years, 3,600 rpm was the norm. As of December 2013, the platters in most consumer-grade HDDs spin at either 5,400 rpm or 7,200 rpm.

Hibernator

Hibernation[2] (or suspend to disk) in computing is powering down a computer while retaining its state. Upon hibernation, the computer saves the contents of its random access memory (RAM) to a hard disk or other non-volatile storage. Upon resumption, the computer is exactly as it was before entering hibernation.

Many systems also support a low-power sleep mode in which the processing functions of the machine are powered down, using a little power to preserve the contents of RAM and support waking up. Instantaneous resumption is one of the advantages of sleep mode over hibernation. A hibernated system must start up and read data back to RAM, which typically takes time. A system in sleep mode only needs to power up the CPU and display, which is almost instantaneous. On the other hand, a system in sleep mode still consumes power to keep the data in the RAM. Detaching power from a system in sleep mode results in data loss, while cutting the power of a system in hibernation has no risk; the hibernated system can resume when and if the power is restored. Both shut down and hibernated systems may consume standby power unless they are unplugged.

Round-Robin scheduling

Round-robin (RR) is one of the algorithms employed by process and network schedulers in computing. As the term is generally used, time slices (also known as time quanta) are assigned to each process in equal portions and in circular order, handling all processes without priority (also known as cyclic executive). Round-robin scheduling is simple, easy to implement, and starvation-free. Round-robin scheduling can also be applied to other scheduling problems, such as data packet scheduling in computer networks. It is an operating system concept.

The name of the algorithm comes from the round-robin principle known from other fields, where each person takes an equal share of something in turn.

Multiple-choice knapsack problem

The knapsack problem or rucksack problem is a problem in combinatorial optimization: Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. It derives its name from the problem faced by someone who is constrained by a fixed-size knapsack and must fill it with the most valuable items.

The problem often arises in resource allocation where there are financial constraints and is studied in fields such as combinatorics, computer science, complexity theory, cryptography, applied mathematics, and daily fantasy sports.

The knapsack problem has been studied for more than a century, with early works dating as far back as 1897.The name ”knapsack problem” dates back to the early works of mathematician Tobias Dantzig (18841956), and refers to the commonplace problem of packing the most valuable or useful items without overloading the luggage.

Round-robin (RR) is one of the algorithms employed by process and network schedulers in computing. As the term is generally used, time slices (also known as time quanta) are assigned to each process in equal portions and in circular order, handling all processes without priority (also known as cyclic executive). Round-robin scheduling is simple, easy to implement, and starvation-free. Round-robin scheduling can also be applied to other scheduling problems, such as data packet scheduling in computer networks. It is an operating system concept.

The name of the algorithm comes from the round-robin principle known from other fields, where each person takes an equal share of something in turn.

3.3 Solid-State Disk bandwidth allocation

Power modelling

A disk has three power phasesseek, idle, and active[6]where the disk seeks during the seek phase, spins without read, write, or seek operations during the idle phase, and actually reads or writes during the active phase. A disk seek phase can consist of four steps: a speedup step, when the arm is accelerated; a coast step, when the arm moves at its maximum velocity for long seeks; a slowdown step, when the arm decelerates to reach the desired track; and a settle step, when the disk controller adjusts the head to access the desired location. The power consumption required for the seek phase includes all these four steps.

The seek phase consumes the most energy. Therefore, reducing the number of seek operations is critical. In order to minimize the number of seek operations, it is advantageous to process low bit-rate requests first from the SSD. During each round T_r , the exact amount of data, $b_i T_r$ needs to be read for video i. For example, consider two videos with bit-rates of 1.5 and 3 Mbps; to read 3Mbits of data, the former requires two seek operations, whereas the latter requires one seek during a round of 1 s. Therefore, we give a higher priority to requests for videos with low bit- rates. Let $R_{k,m}$ be a request for a video cached on the SSD with original files stored on disk k and with the mth lowest bit rate[5]. Then, we have an array of request combinations served from the SSD for disk k, A_k as follows:

$$A_k = \{\varphi, \{\mathrm{R}_{k,1}\}, \{\mathrm{R}_{k,1}, \mathrm{R}_{k,2}\}, \ldots, \{\mathrm{R}_{k,1}, \ldots, \mathrm{R}_{k,Nk-1}\}\}$$

where N_k represents the number of elements in this array. Let $V(j)$ be the video index for request j. We introduce a binary variable $X_{k,m}$ as follows: if videos in the mth combination in an array A_k (e.g., $C_{k,m}$) are served by SSD, then $X_{k,m}$ = 1; otherwise, $X_{k,m}$ = 0. For example, if $X_{k,3}$ = 1, then two requests ($R_{k,1}$ and $R_{k,2}$) are served by SSD.We then calculate the disk

bandwidth utilization at each speed level l when $X_{k,m}$ = 1, $D_{k,m}(l)$ as follows:

$$D_{k,m}(l) = \sum_{j \in C_{k,N_k} - C_{k,m}} \frac{T_s + \frac{b_v T_r}{r(l)}}{T_r} \quad \text{(Equ : 3.4)}$$

1. The total seek time is $\Sigma_{j \in C_{k,N_k} - C_{k,m}} T_s$. Therefore, the energy required in the seek phase, $E^{seek}_{k,m}(l)$, is calculated as:

$$E^{seek}_{k,m}(l) = \sum_{j \in C_{k,N_k} - C_{k,m}} T_s P_s(l) \quad \text{(Equ : 3.5)}$$

2. The total time taken to read data for T_r is $\Sigma_{j \in C_{k,N_k} - C_{k,m}} \frac{b_v T_r}{r(l)}$. Therefore, the energy required for reading data, $E^{active}_{k,m}(l)$, is calculated as:

$$E^{active}_{k,m}(l) = \sum_{j \in C_{k,N_k} - C_{k,m}} \frac{b_v T_r}{r(l)} P_a(l) \quad \text{(Equ : 3.6)}$$

3. If no disk activity is occurring, the disk is rotating without reading or seeking, which requires a power of $P_i(l)$.We calculate the total idle time during a round of length T_r by subtracting the seek and read times from T_r . Thus, the energy required in the idle phase, $E^{idle}_{k,m}(l)$, can be calculated as:

$$E^{idle}_{k,m}(l) = (T_r - \sum_{j \in C_{k,N_k} - C_{k,m}} \frac{b_v T_r}{r(l)}) P_i(l) \quad \text{(Equ : 3.7)}$$

We can now determine the power consumption by disk k when the speed level is l and $X_{k,m}$ D= 1, $P_{k,m}(l)$ as:

$$P_{k,m}(l) = \frac{E^{seek}_{k,m}(l) + E^{active}_{k,m}(l) + E^{idle}_{k,m}(l)}{T_r} \quad \text{(Equ : 3.8)}$$

Redundant Array of Independent Disks architecture

RAID[1] (Redundant Array of Independent Disks, originally Redundant Array of Inexpensive Disks) is a data storage virtualization technology that combines multiple physical disk

drive components into one or more logical units for the purposes of data redundancy, performance improvement, or both.

Data redundancy, although taking up extra space, adds to disk reliability. This means, in case of disk failure, if the same data is also backed up onto another disk, we can retrieve the data and go on with the operation. On the other hand, if the data is spread across just multiple disks without the RAID technique, the loss of a single disk can affect the entire data.

RAID works by placing data on multiple disks and allowing input/output (I/O) operations to overlap in a balanced way, improving performance. Because the use of multiple disks increases the mean time between failures (MTBF), storing data redundantly also increases fault tolerance. Fig.3.1 shows basic RAID architecture.

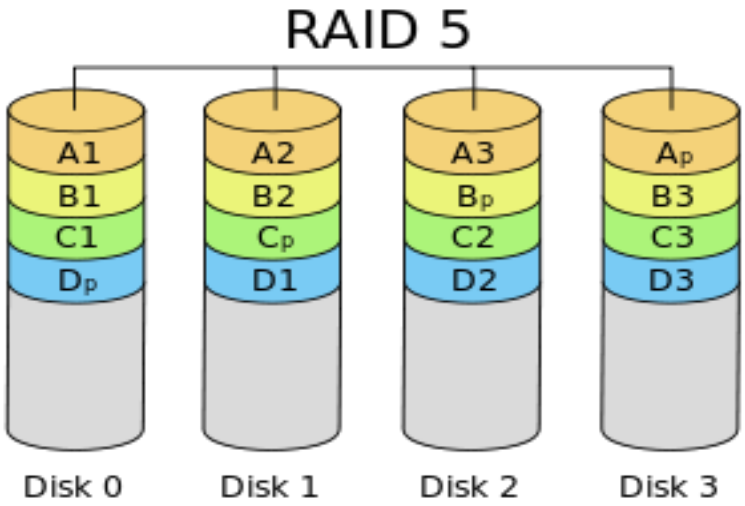


Fig. 3.1: Basic RAID architecture.

A RAID controller can be used as a level of abstraction between the OS and the physical disks, presenting groups of disks as logical units. Using a RAID controller can improve performance and help protect data in case of a crash.

Coping with disk speed changes

A considerable amount of time may be required to transition from one speed to another. Disk requests cannot be handled during this speed transition period, which causes playback jitter. Prefetching data, which requires contingent disk bandwidth, is necessary during the speed transition period to prevent this problem.

Let T_t be the transition time between two different speed levels . The server has to prefetch a sufficient amount of data in T_t s. In order to ensure real-time playback[6], the data to

be consumed during the round immediately after the transition period must also be prefetched. Thus, a total of T_t + T_r s of data can be prefetched for each stream if the speed changes. If the prefetching starts N p rounds before speed transition, then the amount of data for request j, (T_t + T_r)$b_{V(j)}$, needs to be prefetched across N p rounds.

Let $L_{k,m}$ be the speed level selected when $X_{k,m}$ = 1. Because some contingent disk bandwidth may be reserved to prepare for disk speed changes, we select the lowest speed level in $\{l \mid \mathrm{D}_{k,m}(l) + D_{k,m}{}^{cont}(l) \leq 1\}$ for $L_{k,m}$ to maximize the reduction in disk energy consumption. We also introduce a binary variable, ch k,m , to indicate whether speed changes occur in disk k when $X_{k,m}$ = 1. If the selected speed level is the same as the current speed level, then $ch_{k,m}$ = 0; otherwise, $ch_{k,m}$ = 1.

Problem formulation

Let B_{ssd} be the SSD total bandwidth in MB/s. Let $B_{k,m}$ be the SSD bandwidth in MB/s required to serve all of the requests in $C_{k,m}$. Let P_t be the power required for speed transition. We introduce a variable, N_c , to denote the number of simultaneous speed changes. We then have two constraints, (C1 and C2), for SSD bandwidth allocation as follows:

1. **C1**: The total bandwidth needed to serve requests from the SSD must not exceed the SSD bandwidth;

2. **C2**: The number of speed change must not exceed N_c; $\sum_{k=1}^{N_d}\sum_{m=1}^{N_k} \mathrm{X}_{k,m} ch_{k,m} \leq N_c$. This constraint also limits the amount of energy consumption required for the speed transition by $N_c P_t T_t$.

We can then formulate the bandwidth allocation problem (BAP), which minimizes power consumption for determining $X_{k,m}$, (k = 1, . . . ,N_d , and m = 1, . . . , N_k) as:

$$\begin{aligned} \text{Minimize } & \sum_{k=1}^{N_d}\sum_{m=1}^{N_k} \mathrm{X}_{k,m} P_{k,m}(L_{k,m}) \\ \text{subject to } & \sum_{k=1}^{N_d}\sum_{m=1}^{N_k} \mathrm{X}_{k,m} B_{k,m} \leq B_{ssd}, \\ & \sum_{k=1}^{N_d}\sum_{m=1}^{N_k} \mathrm{X}_{k,m} ch_{k,m} \leq N_c, \\ & \sum_{m=1}^{N_k} \mathrm{X}_{k,m} = 1, \\ & X_{k,m} \epsilon 0, 1. \end{aligned}$$

Bandwidth allocation algorithm

BAP is a variant of the multi-dimensional multiple-choice knapsack problem (MMKP)[6], which is NP-hard. In the MMKP, each object consists of a set of items, with each item having a weight and profit. The problem involves selecting exactly one item from each object in order to maximize the total profit while satisfying multiple constraints. We can convert the BAP into the MMKP by considering the SSD as a knapsack and by regarding each array for disk k, Ak as an object. The objective of the BAP is to minimize power consumption while satisfying the two constraints of C1 and C2.

Because BAP is NP-hard, we propose a heuristic solution that runs in polynomial time. In general, greedy algorithms exhibit good performance with multiple-choice knapsack problems. Therefore, we use a greedy approach in which we define parameters denoting the ratio of the power relative to the bandwidth and try to select that with the best ratio. Our solution consists of two phases, in which we consider **C1** alone in the first phase and **C2** step by step in the second phase. For this purpose, we introduce the temporary variables I_k , (k = 1,...N_d) to indicate that the I_k th element in an array of A_k is selected for disk k. details of the bandwidth allocation algorithm (BAA) are presented in Fig. 3.2. For the first phase, we define a series of parameters ${Q_{k,m}}^F$ for each disk k, (k = 1...N_d and m = 1...N_k - 1) as:

$${Q_{k,m}}^F = \frac{P_{k,m}(L_{k,m})}{B_{k,Nk} - B_{k,m}} \qquad \text{(Equ : 3.9)}$$

The values of ${Q_{k,m}}^F$ represent the ratio of power difference to bandwidth difference, and the values with the best ratio are selected one by one.

I_k are all initialized to N_k , which requires the lowest power consumption but the highest SSD bandwidth. Thus, some of the I_k values can be decreased to meet the bandwidth requirement, **C1**. We first select the most profitable change so that the lowest value of ${Q_{k,m}}^F$;L is searched in order to replace the value I_k by L, which minimizes the power consumption while maximizing the decrease in the SSD bandwidth. This operation is repeated until constraint **C1** is satisfied (lines 8-14 in algorithm).

1. List of values: ${Q_{k,m}}^F$: G^F
2. List of values: ${Q_{k,m}}^S$: G^S
3. Temporary variable I_k : (k=1.....N_d)
4. Temporary variable: V and W

5. $X_{k,m}$ are all initialized to 0: (k=1.....N_d and m=1.....N_k)
6. I_k are all initialized to N_k (k=1....N_d)
7. V $\leftarrow \sum_{k=1}^{N_d} B_k I_k$
8. **while** V>1 **do**:
9. Find the lowest value of $Q_{k,L}^{F} \epsilon G^F$ and remove $Q_{k,L}^{F}$ from G^F
10. **if** L $< I_k$ **then**
11. V $\leftarrow V$+B$_{k,L} - B_{k,I}k$
12. $I_k \leftarrow L$
13. **end if**
14. **end while**
15. W $\leftarrow \sum_{k=1}^{N_d} ch_k I_k$
16. **while** W$> N_c$ **do**:
17. Find the lowest value of $Q_{k,L}^{S} \epsilon G^S$ and remove $Q_{k,L}^{S}$ from G^S
18. **if** V $\leftarrow V$+B$_{k,L} - B_{k,I}k \leq B_{ssd}$ and $ch_{I,k}$ = 0 then
19. V $\leftarrow V$+B$_{k,L} - B_{k,I}k \leq B_{ssd}$
20. $W \leftarrow W - 1$
21. $I_k \leftarrow L$
22. **end if**
23. **if** $G^S = \phi$ **then**:
24. Break the loop
25. **end if**
26. **end while**
27. for k=1 to N_d do

28. $X_{k,Ik} \leftarrow 1$

29. end for

In the second phase, if condition **C2** does not hold, then some of the I_k values may be changed to reduce the number of speed changes. Thus, we establish new parameters $Q_{k,m}^{S}$ as follows:

$$Q_{k,m}^{S} = P_{k,m}(L_{k,m}) - P_k I_k(L_{k,I,k}) \qquad \text{(Equ : 3.10)}$$

which are only defined when $ch_{k,m} = 0$ and $ch_{k,Ik} = 1$. We find the lowest value, $Q_{k,L}^{S}$;L, from a set of $Q_{k,m}^{S}$ values and change the value of I_k to L if $ch_{k,}L = 0$. This minimizes the increase in power consumption while the speed of the disk k remains unchanged. These steps are repeated until constraint **C2** is satisfied (lines 16-26 in algorithm).

The BAA runs when a new client is accepted for video service in order to accommodate new workload changes immediately. However, executing the BAA may require that the speeds of some disks change[4], resulting in frequent speed changes in the long term. Thus, speed changes are allowed only periodically in order to prevent this problem, where with every αth execution of the BAA, the value of N_c is set to an upper bound value, U, (U $>$ 0); otherwise, N_c is set to 0.

Performance evaluation

The disk had a maximum data transfer rate of 170 MB/s and a typical seek time of 8.5 ms. However, we considered a range of up to five speeds between 2880 and 7200 RPM in order to analyze the effects of various speed configurations.

We compared the BAA with two other bandwidth allocation methods based on our model as follows:

(a) Lowest bit-rate first selection (LS): As described in Subsection V-A, for disk power reduction, it is advantageous to serve low bit-rate requests from the SSD. Thus, the LS scheme selects the request with the lowest bit-rate first, subject to the SSD bandwidth limitation.

(b) Uniform selection (US): Suppose that I_k is an index of the selected element in A_k . The US scheme increments the I_k values of all the disks one by one until the SSD

bandwidth limit is not exceeded.

We considered a disk array consisting of 50 multi-speed disks, each of which corresponded to one of the following five sets of RPM configurations: 2880, 7200, 5040, 7200, 2880, 5040, 7200, 2880, 5040, 6120, 7200, and 2880, 3960, 5040, 6120, 7200. Unless stated otherwise, in a Zipf distribution, was set to 0.271, which was measured for a real VoD application. Videos typically last between 1 and 2 h. Therefore, the length of each video was selected randomly from this range.

A server was assumed to serve 1000 video files with ran- domly chosen bit rates between 10.4 and 20.8 Mb/s, which is typical of HD quality videos. The idle power of the SSD was 0.03 W, whereas power for the SSD read operation was 2.1 W, based on real measurements . The round length was 2 s and N_p was 12 s . The speed transition time was 6 s and the transition power was 21 W . The arrival of client requests was modeled as a Poisson process with an average arrival rate for requests of 0.4/s. U was set to 1. We profiled the disk power consumption over all the disks for 10 h.

A server was assumed to serve 1000 video files with randomly chosen bit rates between 10.4 and 20.8 Mb/s, which is typical of HD quality videos. The idle power of the SSD was 0.03 W, whereas power for the SSD read operation was 2.1 W, based on real measurements . The round length was 2 s and N p was 12 s . The speed transition time was 6 s and the transition power was 21 W . The arrival of client equests was modeled as a Poisson process with an average arrival rate for requests of 0.4/s. U was set to 1. We profiled the disk power consumption over all the disks for 10 h.

Effect on Solid State Disk size

Fig. 3.2 shows the variations in disk power consumption with each algorithm versus the SSD size when B_{ssd} = 1 GB/s, where the results indicate the following:

(a) BAA always performed the best. Specifically, it reduced the power consumption from 10 to 13% compared to the LS scheme and from 10 to 12% compared to the US scheme. To support the high request rates for popular video files stored on the cache, all the algorithms attempted to serve as many requests from the SSD as possible, allowing the algorithm to be executed until SSD bandwidth could be utilized fully. However, BAA performed better than the other schemes because it chose disk speed to minimize the total disk power consumption.

(b) When using the BAA and US schemes, the reduction in power consumption increased with the SSD size. However, the effect of the SSD size was negligible for the LS scheme.

(c) The difference in power consumption between the BAA and the other algorithms increased with the SSD size, suggesting that the effect of the BAA was more pronounced, especially when many videos were stored on the SSD.

(d) The difference in the power consumption between US and LS schemes was below 1%.

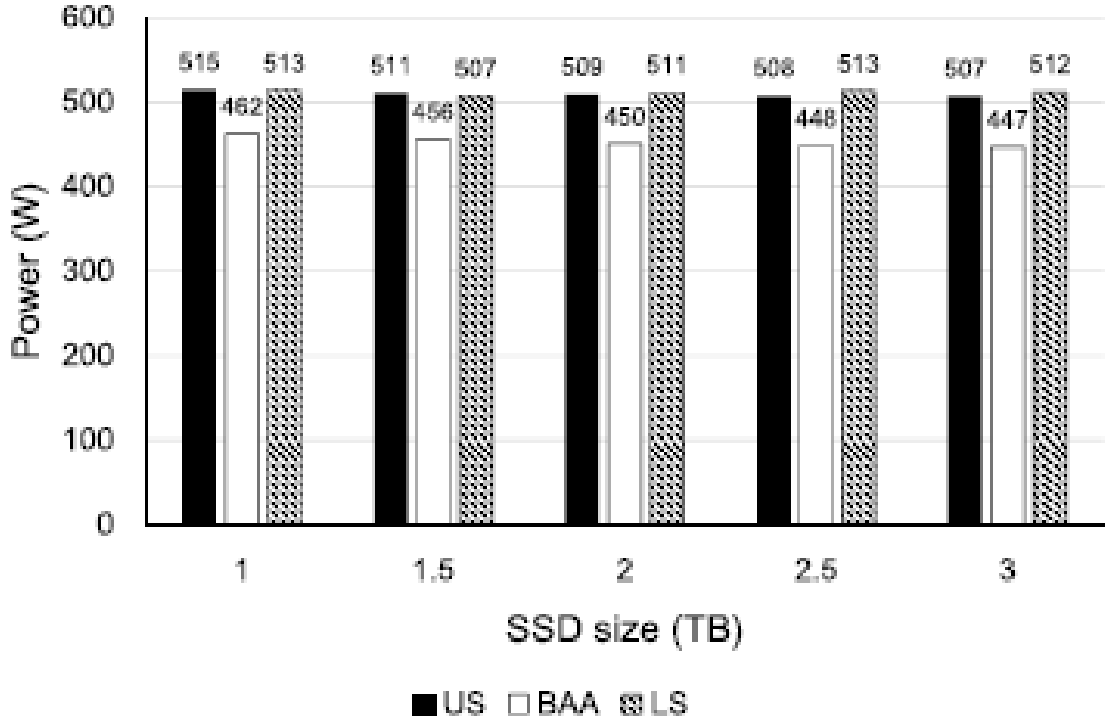


Fig. 3.2: Disk power consumption against SSD size.

Effect on Solid State Disk bandwidth

Fig. 3.3 shows the disk power consumption versus the SSD bandwidth when S ssd = 2.5 TB, where the results indicate the following:

(a) BAA always performed the best. Specifically, it reduced the power consumption from 11 to 14% when compared to the other two schemes. If the storage accounts

for 40% of the total cloud power consumption , this reduction corresponded to a reduction of 4.4 to 5.6% of total power consumption.

(b) The reduction in the disk power consumption increased with the SSD bandwidth, which allowed more requests to be served by the SSD cache.

(c) The difference in power consumption between the BAA and the other schemes was maximized when the SSD bandwidth was moderate (B ssd = 1.25 GB/s and B_{ssd} = 1.5 GB/s). With sufficient SSD bandwidth, the lowest speeds could be selected for most of the disks, whereas most of the disks required the highest speeds to handle the high demand for disk requests when the SSD bandwidth was low. However, when the bandwidth was moderate, the selection of the disk speed had a tremendous effect on power consumption. The BAA selected the speed levels with the aim of minimizing power consumption, thereby obtaining greater power savings compared with the other schemes, particularly when the bandwidth was moderate.

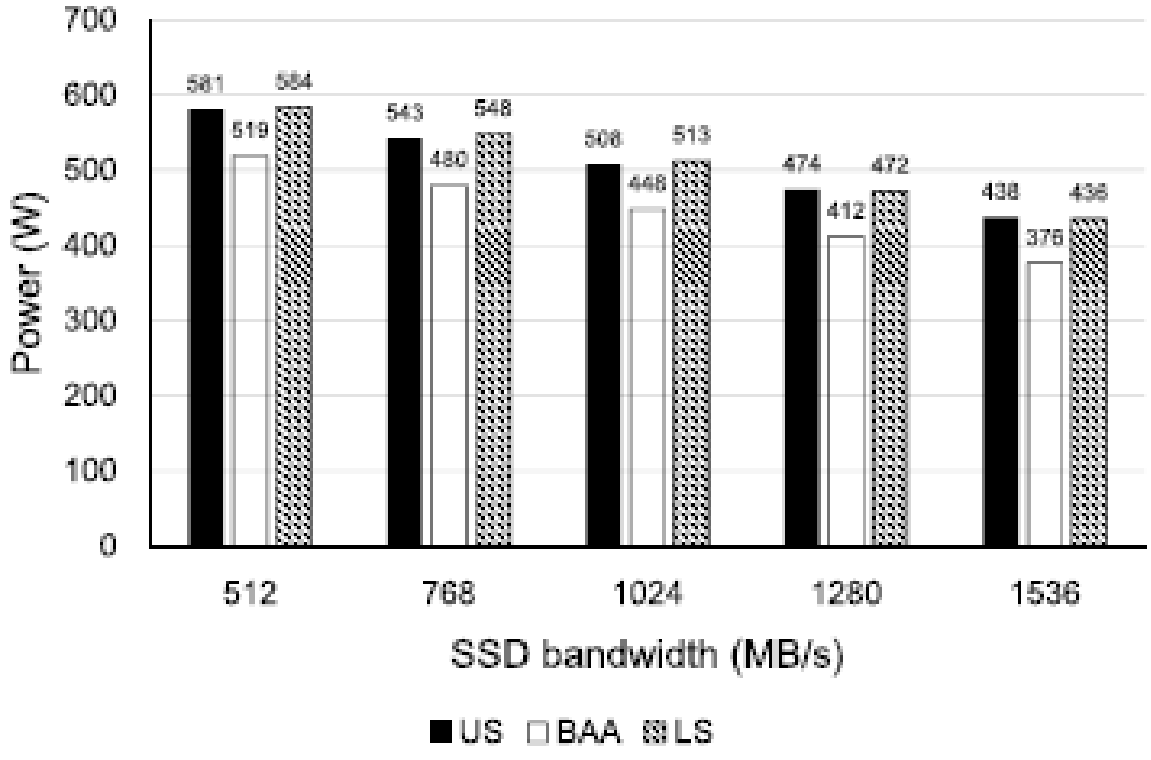


Fig. 3.3: Disk power consumption against SSD bandwidth.

The first come, first served (FCFS) policy that schedules the video with the longest outstanding request can perform better than the maximum queue length (MQL)

policy that chooses the video with the maximum number of outstanding requests. Additionally, multicasting is better exploited by scheduling playback of the n most popular videos at predetermined, regular intervals (hence, termed FCFS-n). If user reneging can be reduced by guaranteeing that a maximum waiting time will not be exceeded, then performance of FCFS-n is further improved by selecting the regular playback intervals as this maximum waiting time. For an empirical workload, we demonstrate a substantial reduction (of the order of 60%) in the required server capacity by batching.

Conclusion

A new power management scheme for a video server was developed in which an SSD is used as a cache for a disk array with multi-speed disks. First, how the power consumption at each disk speed level varies was examined with the amount of SSD bandwidth allocated to each disk. Then formulated an optimization problem with the purpose of minimizing the overall disk power consumption subject to the SSD bandwidth while limiting the number of speed changes. Then a two-phase algorithm was proposed to solve this problem. The first step was to find the best configuration of the SSD bandwidth allocation for each disk in order to determine the requests that can be served by the SSD. The second step was to ensure that the number of disk speed changes is limited. The experimental results showed that the proposed BAA could reduce the power consumption by between 10 and 15% as compared to two standard algorithms that we derived from our model. The results also demonstrated that: 1) the effectiveness of the proposed scheme is pronounced with a large SSD size, adequate SSD bandwidth, and skewed access patterns; and 2) a trade-off exists between variations in power consumption and disk speed changes based on the number of times the algorithm is executed. The results obtained provide useful guidelines for constructing power-efficient video servers by employing an SSD cache and multi-speed disks. Frequent cache replacements produce many write operations, thus causing the SSD to wear out at a quicker rate. One possible method is to use a throttling technique by limiting the number of cache replacements with the aim of maximizing cache hit ratio. The scheme can be used to support various types of storage architecture, including hot and cold disk farms.

REFERENCES

[1] J. E. Pinheiro and R. Bianchini, "Energy conservation techniques for diskarray-based servers," in Proc. ACM/IEEE Conf. Supercomput., Jun. 2004, pp. 88-95.

[2] S. Gurumurthi, "Power management of enterprise storage systems," Ph.D. dissertation, Dept. Comput. Sci. Eng., Pennsylvania State Univ., State College, PA, USA, 2005.

[3] M. Kim and M. Song, Saving energy in video servers by the use of multispeed disks, IEEE Trans. Circuits Syst. Video Technol., vol. 22, no. 4, pp. 567580, Apr. 2012.

[4] C. Li, P. Shilane, F. Douglis, H. Shim, S. Smaldone, and G. Wallace, Nitro: A capacity-optimized SSD cache for primary storage, in Proc. USENIX Annu. Tech. Conf., Jun. 2014, pp. 501512.

[5] J. Hui, X. Ge, X. Huang, Y. Liu, and Q. Ran, E-HASH: An energy- efficient hybrid storage system composed of one SSD and multiple HDDs, in Proc. Int. Conf. Swarm Intell., Jun. 2012, pp. 527534. Trans. Circuits Syst. Video Technol., vol. 28, no. 4, pp. 984996, Apr. 2018.

[6] C. Chen, D. Xiong, K. Zhao, C. W. Chen, and T. Zhang, Realizing low- cost flash memory based video caching in content delivery systems, IEEE Trans. Circuits Syst. Video Technol., vol. 28, no. 4, pp. 984996, Apr. 2018.

[7] Q. Zhu and Y. Zhou, "Power-aware storage cache management," IEEE Trans. Comput., vol. 54, no. 5, pp. 587-602, May 2005.