Impact of Flash Memory on Video-on-Demand Storage: Analysis of Tradeoffs

Moonkyung Ryu
College of Computing
Georgia Institute of Technology
Atlanta, GA, USA
mkryu@gatech.edu

Hyojun Kim
College of Computing
Georgia Institute of Technology
Atlanta, GA, USA
hyojun.kim@cc.gatech.edu

Umakishore Ramachandran
College of Computing
Georgia Institute of Technology
Atlanta, GA, USA
rama@cc.gatech.edu

ABSTRACT

There is no doubt that video-on-demand (VoD) services are very popular these days. However, disk storage is a serious bottleneck limiting the scalability of a VoD server. Disk throughput degrades dramatically due to seek time overhead when the server is called upon to serve a large number of simultaneous video streams. To address the performance problem of disk, buffer cache algorithms that utilize RAM have been proposed. Interval caching is a state-of-the-art caching algorithm for a VoD server. *Flash Memory Solid-State Drive (SSD)* is a relatively new storage technology. Its excellent random read performance, low power consumption, and sharply dropping cost per gigabyte are opening new opportunities to efficiently use the device for enterprise systems. On the other hand, it has deficiencies such as poor small random write performance and limited number of erase operations. In this paper, we analyze tradeoffs and potential impact that flash memory SSD can have for a VoD server. Performance of various commercially available flash memory SSD models is studied. We find that low-end flash memory SSD provides better performance than the high-end one while costing less than the high-end one when the I/O request size is large, which is typical for a VoD server. Because of the wear problem and asymmetric read/write performance of flash memory SSD, we claim that interval caching cannot be used with it. Instead, we propose using file-level Least Frequently Used (LFU) due to the highly skewed video access pattern of the VoD workload. We compare the performance of interval caching with RAM and file-level LFU with flash memory by simulation experiments. In addition, from the cost-effectiveness analysis of three different storage configurations, we find that flash memory with hard disk drive is the most cost-effective solution compared to DRAM with hard disk drive or hard disk drive only.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Design Studies; C.5.5 [**Computer Systems Organization**]: Computer System Implementation—*Servers*

MMSys'11, February 23–25, 2011, San Jose, California, USA.
ACM 978-1-4503-0517-4/11/02.

General Terms

Design, Performance

Keywords

Flash Memory, Solid-State Drive, SSD, Video-on-Demand, VoD, Interval Caching

1. INTRODUCTION

Video-on-demand (VoD) services like YouTube [5], Hulu [2], and Netflix [3] have achieved huge success recently. There are several technological factors that have contributed to the success of VoD: (a) high speed networks, (b) data compression algorithms, and (c) powerful CPUs. Due to the success of VoD, the demand for such services is on the rise both in terms of the number of requests as well as the quality (i.e., video bitrate). Larger number of video requests and higher video bitrate impose a larger bandwidth requirement on the storage subsystem. A pure disk-based storage subsystem is a serious impediment to meeting this increased bandwidth requirement in a VoD server architecture.

While the capacity of magnetic disks has been steadily increasing over the past few decades, the data access time has not kept pace with other computing components such as the CPU and memory. If anything, the speed gap between the main memory (DRAM) and the hard disk has only widened. This is not very surprising since the data access time for the disk is limited by the electromechanical delays in the form of seek time and rotational latency. These delays significantly degrade the VoD server performance when serving a number of video requests simultaneously. Figure 1 demonstrates the poor performance of a hard disk drive (HDD) when it services a large number of sequential streams. In this experiment, we have used *xdd* [4] benchmark on the raw device. The file cache of the operating system is disabled, and the I/O request size is 1MB. The disk drive is SEAGATE Cheetah 15K.6, an enterprise class disk with a rotational speed of 15000RPM and a Serial-Attached SCSI (SAS) interface. The flash memory SSD is SAMSUNG MMDOE56G5MXP-0VB, a consumer class one. As shown in the graph, when the number of streams exceeds 20, the aggregate read throughput of the disk drops significantly. On the other hand, the aggregate read throughput of the flash memory SSD remains constant even with 400 streams. When a VoD server services a large number of video streams simultaneously, it generates random read I/O pattern to storage. Therefore, the random

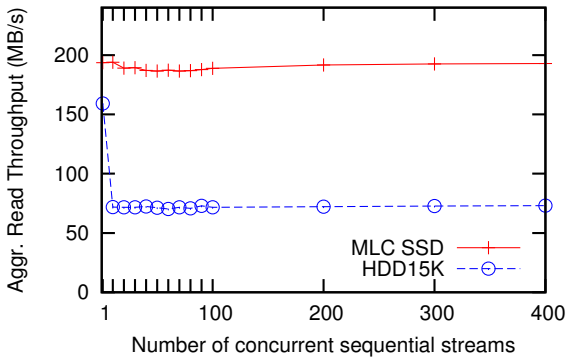


Figure 1: Reading sequential files, 1MB I/O request size. Aggregate read throughput drops quickly from 160MB/s to 70MB/s with 15000RPM hard disk drive. Flash memory SSD provides constant read throughput with a large number of sequential streams.

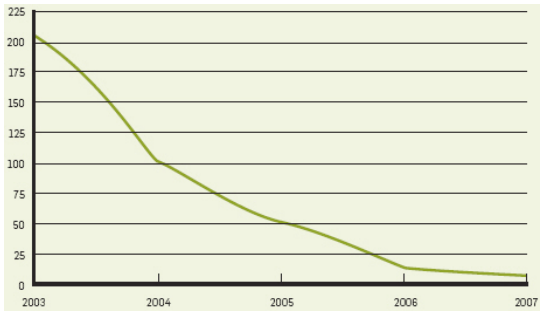


Figure 2: Flash memory $/GB trend

read throughput of a storage device determines scalability of the device, i.e., the number of concurrent video streams that the device can serve.

To serve video streams at a cheaper cost, buffer cache algorithms that use RAM as a buffer cache have been proposed. An efficient buffer cache algorithm can reduce the service latency due to the fast access speed of RAM and can serve more video streams compared to the alternative of adding more disks. Interval caching [8, 9], which is a state-of-the-art buffer cache algorithm for a VoD server, exploits temporal locality when independent requests access the same video file by caching intervals between the successive streams. The average size of intervals and the capacity of the buffer determines the number of concurrent streams that can be served by interval caching. Interval caching provides cheaper cost for serving a video stream in certain range of RAM capacity compared to adding more disks depending on the inter arrival time of video requests.

Flash memory SSD opens up new opportunities for providing a more cost-effective solution for a VoD server. Solid-State Drive (SSD) is a new storage device that is comprised of semiconductor memory chips (e.g., DRAM, Flash memory, Phase change memory) to store and retrieve data rather than using the traditional spinning platters, a motor, and moving heads found in conventional magnetic disks. The term "solid-state" means there are no moving parts in accessing data on the drive. Due to the absence of the mechanical components, the SSD has several benefits such as high speed data access, light weight, low power consumption, no vibration, and no noise.

Among various types of SSDs, flash memory SSD nowadays is rapidly penetrating into modern computer systems. NAND flash memory densities have been doubling since 1996 consistent with Hwang's flash memory growth model, and it is expected to continue until 2012 [14]. Figure 2 shows that the cost trend of the flash memory conforms to his estimation [20]. The sharply dropping cost per gigabyte is what has brought flash memory SSD to the forefront in recent years. Flash-based storage devices are now considered to have tremendous potential as a new storage medium for enterprise servers [13]. The advantages of flash memory SSD are fast random read, low power consumption, and no vibration or noise. On the other hand, its high cost per gigabyte compared to magnetic disks, poor small random write performance, and wear are major concerns compared to the hard disk [12].

Compared with disks, flash memory SSD boasts excellent features that are appropriate for a VoD storage. First, it has very low latency in retrieving data. The access latency of magnetic disks is 10ms. On the other hand, the read access latency of flash memory SSD is 0.1ms [25]. Lower read latency equates to lower delay users experience when starting a video. Second, its random read throughput can reach the sequential read throughput, which is the maximum read throughput, when the I/O request size is large and a multiple of the block size of the flash memory. A VoD server typically uses a large I/O request size ($>$ 1MB). Larger random read throughput means larger numbers of simultaneous video streams that can be serviced by a VoD server. Third, the video popularity pattern is highly skewed and is modeled by a Zipf distribution [26]. This means only a small fraction of the total video repository is accessed most of the time. The highly skewed access pattern gives an opportunity to the flash memory SSD to be efficiently used even though its cost per gigabyte is more expensive than that of a magnetic disk.

Magnetic disk offers capacity while DRAM offers speed of access. The combination has been successful for implementing memory hierarchies to support traditional workload of computer systems including virtual memory and file servers. However, VoD server's requirement is unique in that the storage device needs to provide both capacity and speed. Flash memory fits this requirement since it provides both at a price point that is more favorable than DRAM. Therefore, our idea is to use Flash-based SSD as a buffer cache similar to how interval caching uses DRAM for the same purpose. The purpose of the paper is to investigate the efficacy of Flash-based SSD as an alternative to DRAM as a buffer cache both from the point of view of performance and cost-effectiveness.

The unique contributions of our work are as follows:

(a) We measure and compare performance of various kinds of flash memory SSDs. Surprisingly, we observe that cheap low-end flash memory SSD provides better performance for VoD storage than the expensive high-end flash memory SSD. The low-end flash memory SSD has similar random read throughput to that of the high-end flash memory SSD when I/O request size is large as is typical for a VoD server. The low-end flash memory SSD has less variance in the through-

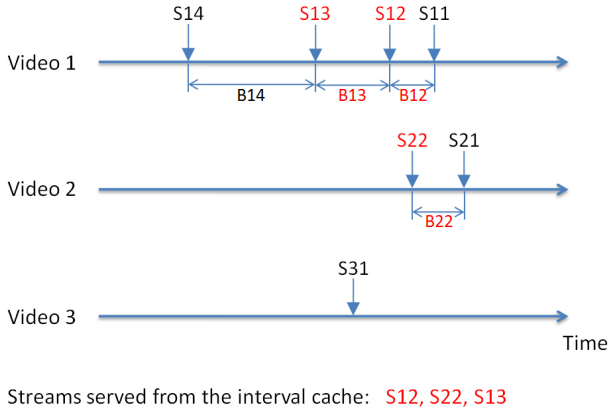


Figure 3: Interval Caching exploits temporal locality accessing the same video and buffers the interval between two successive streams. A preceding stream (e.g., S11) feeds data to a following stream (e.g., S12).

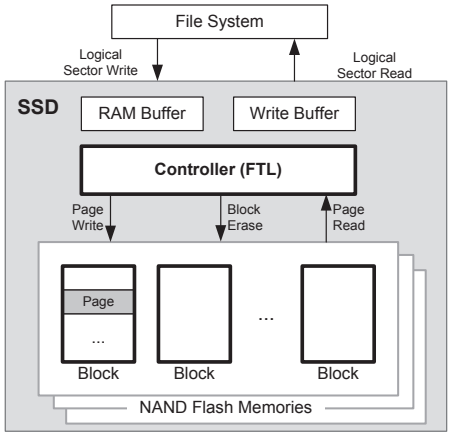


Figure 4: SSD, FTL and NAND flash memory. *FTL emulates sector read and write functionalities of a hard disk allowing conventional disk file systems to be implemented on NAND flash memory*

put when large random read operations and large sequential write operations co-exist. The low-end flash memory SSD has a lower cost per gigabyte than the high-end one. The reason for this is that high-end flash memory SSD adopts more expensive flash memory cells, a complex FTL algorithm, and a larger RAM buffer to improve small random write performance. Because a VoD server uses a large I/O request size and most of its work is read operations from the storage device, we can avoid small random writes to the flash memory SSD in the VoD server.

(b) Unlike RAM, we observe that flash memory SSD is not an appropriate device for the interval caching algorithm due to the asymmetry of read and write access speeds, the unpredictability of write performance incurred by garbage collection, and the limited number of erase operations. Instead, we propose to use file-level Least Frequently Used (LFU) with the flash memory SSD due to the real-time requirement of video service (i.e., file-level) and the highly skewed access pattern of the VoD workload (i.e., LFU).

(c) We compare the performance of RAM and flash memory SSD over a broad range of device parameters and workload parameters by simulation when interval caching is used with RAM and file-level LFU is used with flash memory SSD. In addition, we analyze the cost-effectiveness of three different storage configurations, which are HDD only, DRAM with HDD, and flash memory with HDD. From the analysis, we find that flash memory with HDD is surprisingly cheaper than other two by a factor of 2.

The rest of the paper is organized as follows. Section 2 explains background about interval caching and flash memory SSD. Section 3 summarizes requirements for a VoD storage device. In Section 4, we measure performance of 3 different SSD models and observe that low-end SSD model meets the VoD storage requirements very well. We discuss how to utilize flash memory SSD for a VoD server in Section 5. Section 6 presents simulation experiment results comparing interval caching with RAM and file-level LFU with flash memory. Moreover, we analyze the cost-effectiveness of flash memory as a VoD storage device. Related work is presented in Section 7 and the final section concludes this paper.

2. BACKGROUND

2.1 Interval Caching

The main idea behind the interval caching algorithm [8, 9] is to choose intervals between two consecutive streams watching the same video. The first stream of an interval is referred to as the *preceding* stream, and the second stream is referred to as the *following* stream. The chosen intervals are cached for serving the "following" streams from the buffer cache. Figure 3 illustrates this. The arrows marked by S_{11} through S_{31} represent the temporal pointers corresponding to the distinct streams on videos 1, 2, and 3. Two streams, S_i and S_j are defined to be consecutive if S_j is the stream that next reads the video blocks that have just been read by S_i. For instance, in Figure 3, (S_{11}, S_{12}), (S_{12}, S_{13}), and (S_{13}, S_{14}) form three consecutive pairs for video 1. The interval caching algorithm orders all intervals according to the memory requirements in an increasing order to allocate memory to as many intervals as possible. The memory requirement of an interval is the length of the interval in seconds times video bitrate of the interval involved. For example, Figure 3 shows intervals B12, B22, B13 can be buffered in the interval cache, while there is no more room to buffer B14. When an interval is cached, the preceding stream writes video blocks to the allocated buffer, and the following streams read the blocks from the buffer avoiding disk access. Therefore, continuous read and write operations are involved in the buffer cache.

2.2 Flash Memory Solid-State Drive

Flash memories, including NAND and NOR types, have a common physical restriction, namely, they must be erased before being written [21]. In flash memory, the amount of electric charges in a transistor represents 1 or 0. The charges can be moved both into a transistor by write operation and out by an erase operation. By design, the erase operation, which sets a storage cell to 1, works on a bigger number of storage cells at a time than the write operation. Thus, flash memory can be written or read a single page at a time, but it has to be erased in an erasable-block unit. An erasable-block consists of a certain number of pages. In NAND flash

Device	Seq. Read Throughput	Random Read Throughput	Cost per Gigabyte
HDD15K	160 MB/s	70 MB/s	$1.23
DRAM	> 20 GB/s	> 20 GB/s	$23
MLC SSD	> 155 MB/s	> 155 MB/s	$1.88

Table 1: Characteristics of different storage devices. SEAGATE Cheetah 15K.6 and RiData NSSD-S25-64-C06MPN are measured for the throughput of HDD15K and MLC SSD by *xdd* benchmark program when I/O request size is 1MB. The throughput of MLC SSD is largely different depending on the vendor and the model. Throughput of DRAM comes from DDR3 SDRAM device specification.

memory, a page is similar to a HDD sector, and its size is usually 2 KBytes.

Flash memory also suffers from a limitation on the number of erase operations possible for each erasable block. The insulation layer that prevents electric charges from dispersing may be damaged after a certain number of erase operations. In single level cell (SLC) NAND flash memory, the expected number of erasures per block is 100,000 and this is reduced to 10,000 in two bits multilevel cell (MLC) NAND flash memory.

An SSD is simply a set of flash memory chips packaged together with additional circuitry and a special piece of software called Flash Translation Layer (FTL) [6, 15, 16, 24]. The additional circuitry may include a RAM buffer for storing meta-data associated with the internal organization of the SSD, and a write buffer for optimizing the performance of the SSD. The FTL provides an external logical interface to the file system. A sector is the unit of logical access to the flash memory provided by this interface. A page inside the flash memory may contain several such logical sectors. The FTL maps this logical sector to physical locations within individual pages [6]. This interface allows FTL to emulate a HDD so far as the file system is concerned (Figure 4).

Agrawal et al. enumerate the design trade-offs of SSDs in a systematic way, from which we can get a good intuition about the relation between the performance of SSD and the design decisions [6]. However, the fact of the matter is that without the exact details of the internal architecture of the SSD and the FTL algorithm, it is very difficult to fully understand the external characteristics of SSDs [7].

Nevertheless, at a macro level we can make two observations about SSD performance. First, they show their best performance for sequential read/write access patterns. Second, they show the worst performance for random write patterns.

More complicated FTL mapping algorithms with more resources have been proposed to get better random write performance [24]. However, due to the increased resource usage of these approaches, they are used usually for high-end SSDs. Even though high-end SSDs using fine grained FTL mapping schemes can provide good random write performances, the effect of background garbage collection will be a problem considering the soft real-time requirements of VoD server systems.

3. VIDEO-ON-DEMAND STORAGE

Video data is classified as *continuous media* (CM) data because it consists of a sequence of media *quanta* (e.g., video frames), which is useful only when presented in time [11]. A VoD server differs significantly from other types of servers that support only textual data because of two fundamental CM characteristics.

• Real-time retrieval: CM data should be delivered before the data becomes meaningless. Failure to meet this real-time constraint leads to jerkiness or hiccups in video display.

• High data transfer rate and large storage space: A digital video playback demands high bitrate (e.g., 2˜20Mbps for MPEG4). The size of a video is determined by the bitrate of the video and the length of the video, therefore, a video requires a large amount of capacity from the storage device. In practice, a VoD server must handle playback requests for several streams simultaneously. Even when multiple streams access the same file (such as a popular file), different streams might access different parts of the file at the same time. Therefore, a number of sequential read streams generates random read operations to the storage device. To serve a large number of high bitrate streams simultaneously, the storage device of the VoD server should support a large amount of random read bandwidth. Moreover, the storage device should have a large amount of capacity to store a number of video files that are large in size.

In summary, VoD storage should have both large bandwidth and large capacity for a scalable VoD server. Table 1 shows the random read throughput and cost per gigabyte of three different storage devices, Disk, DRAM, and flash memory SSD. Disk is a capacity optimized device. On the other hand, DRAM is a bandwidth optimized device. Flash memory SSD provides a balanced combination of bandwidth and capacity. This characteristic of the flash SSD gives us an opportunity to efficiently exploit the device for VoD storage.

4. MEASUREMENT OF FLASH SSDS

Different SSDs show very different performance characteristics because SSD manufacturers use different types of NAND flash memories, internal architectures, and FTL algorithms considering design trade-offs and target users [6].

In this section, we show our measurement results with various SSDs. In VoD storage, the workload mainly consists of large but scattered read requests to service multiple streaming clients. Therefore, large random read throughput is the most important metric of SSDs. In addition, when the flash memory SSD is used as a buffer cache, large sequential write throughput is also meaningful because we may want to change the contents of the SSD to accommodate changing popularity of video files. Another important evaluation criterion is deterministic response times. All SSDs have very complicated software layer, which maintains the mapping from logical sectors to physical location within SSDs due to the very nature of NAND flash memory that mandates writing in units of large erasable-blocks rather than individual sectors. The background garbage collection and wear leveling activities can make response time irregular. A desirable SSD for use in a VoD server should have high throughput for large random reads and large sequential writes, and the throughput also needs to be stable.

4.1 Experiment Setting

We have used *xdd* benchmark to measure read/write through-

	A	B	C	D
MODEL	**RiData NSSD-S25-64-C06MPN**	**SAMSUNG MMDOE56G5MXP-0VB**	**INTEL X25-M G1**	**Fusion-io ioDRIVE 80GB SLC**
CLASS	LOW	LOW	HIGH	ENTERPRISE
TYPE	MLC	MLC	MLC	SLC
CAPACITY	64GB	256GB	80GB	80GB
$/GB	$1.88	$2.34	$2.50	$61.95

Table 2: Characteristics of four flash memory SSDs. SSD A, B, and C are classified as HIGH or LOW based on the small random write performance of the SSDs. We do not measure the enterprise class SSD.

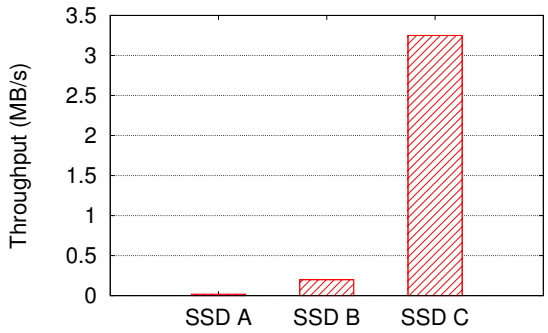


Figure 5: Small random write throughput of different SSDs. 4KB request size is used.

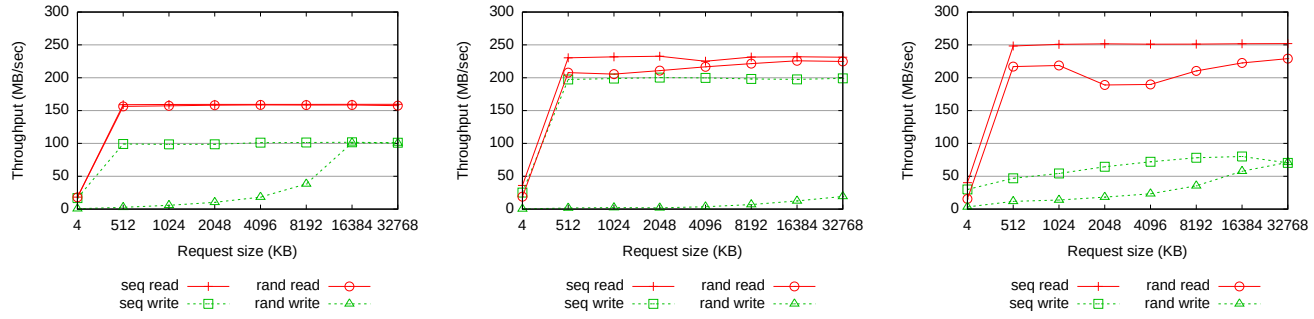
put of SSDs on various request sizes. The benchmark generates random (but aligned) read/write requests with given test size, and reports average throughput. We have compared the performance of three SSDs, which are fairly inexpensive. Intentionally, we do not consider enterprise class SSDs, which are extremely expensive. These enterprise class SSDs provide high reliability and very good random write performance to target enterprise storage server systems. However, such attributes are not important for building a cost-effective VoD server for reasons of the workload characteristics that it has to deal with (see Section 3). Table 2 lists the four SSDs, and some important parameters: cell type, capacity, and price per gigabytes. We classify the three SSDs we experiment with, namely, A, B, and C as HIGH or LOW based on the small random write performance of the SSDs. Figure 5 shows the small random write performance of the three SSDs when the request size is 4KB.

4.2 SSD Throughput Results

Figure 6 shows the throughput measurement results with the three different SSDs for a variety of workloads. Y-axis shows measured average throughput, and higher value means better performance. Test request sizes are shown on the X-axis. As we already explained, large random read and large sequential write are important for the VoD system. In most SSDs and certainly for the ones we experimented with as can be seen from Figure 6, the random read throughput comes close to the sequential read throughput when the request size is more than or equal to 512KB. For SSDs A and C, the random write throughput approaches the throughput of the sequential write workload when the request size is more than or equal to 16MB. However, SSD B has a serious performance problem for random write operations even when the request size is large.

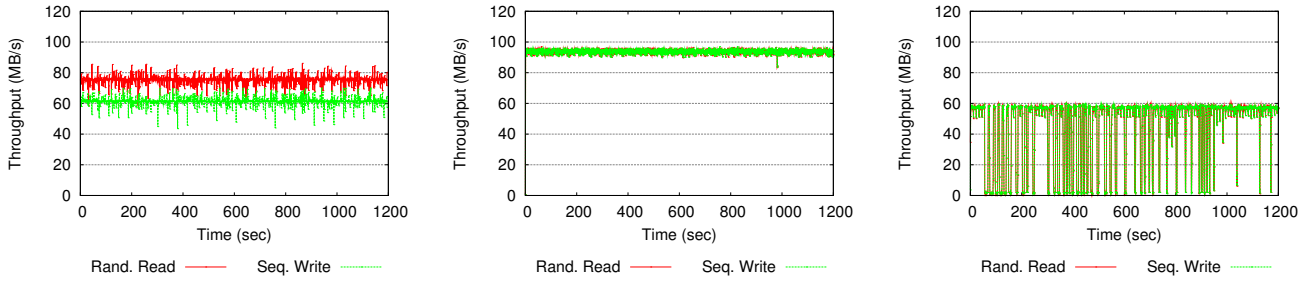
Our real interest in understanding the performance of SSDs for the workload of interest from the point of the VoD server is, namely, large random reads simultaneous with sequential writes. Because the flash memory is intended to be used as a buffer cache in the VoD server, write operations will be replacing unpopular video files while read operations serve the video streams. To figure out how the random read performance is affected in the presence of write operations, we measure the throughput of the SSDs for the combined workload (i.e., Random read and Sequential write operations). We use 1MB request size for the measurement, and we run the experiment for 20mins. Figure 7 shows the time series of the random read throughput and the sequential write throughput of SSD A, SSD B, and SSD C respectively. Figure 8 shows the mean and the standard deviation of the random read throughput of the SSDs for the same combined workload as in Figure 7. Surprisingly, the high-end SSD C that has the best small random write performance (Figure 5) and very good random read performance (Figure 6) shows the worst random read throughput for the combined workload, and worse yet it has very large standard deviation from the mean throughput. On the other hand, the low-end SSDs (i.e., SSD A and B) have very small deviation from the mean throughput. This result reveals a very important insight. As we have already mentioned earlier (see Section 2.2), high-end SSDs incorporate sophisticated algorithms for garbage collection and wear-leveling in the FTL layer to boost random (small) write performance. It is impossible to predict from the host side when such algorithms may be invoked internally in the device, and how these algorithms may interact with other requests coming simultaneously to the device. These are the factors most likely adversely affecting the random read performance for SSD C (a high-end SSD) when write operations are requested simultaneously even though the write operations are sequential with large request sizes.

It is not possible to explain the exact reason why the random read performance is heavily influenced by write operations without the internal design details of SSD products. However, we can find probable causes from already published literature. Feng et al. [7] have shown that read response time can be delayed by procrastinated write operations, and STEC [1] has reported that write response time of consumer level SSD can vary significantly due to the background wear leveling and block relocation activities. Based on the good small random write performance of SSD C, we can cautiously guess that SSD C is using fine-grained mapping FTL algorithm internally. Fine-grained mapping



(a) SSD A **(b)** SSD B **(c)** SSD C

Figure 6: Throughput of various SSDs with different I/O request size



(a) SSD A **(b)** SSD B **(c)** SSD C

Figure 7: Throughput of SSDs for combined workload: Random read and Sequential write (1MB request size is used). For SSD B and C, red line is overlaid with green line because random read throughput and sequential write throughput vary in similar values.

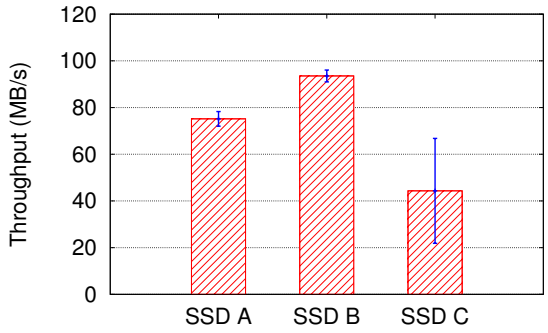


Figure 8: Mean and standard deviation of the random read throughput for the same combined workload as in Figure 7.

scheme may provide good random write performance but requires efficient garbage collection, which can make response time irregular. The result of Figure 8 is well matched with our guess. On the other hand, coarse grained mapping FTL algorithms show poor small random write performance, but do not require background garbage collection process in general. Instead, they may use much simpler merge operations whose overhead is minimal for sequential writes. The stable read performance observed for SSD A and B in Figure 8 can be well explained if SSD A and B (being low-end SSDs) are using coarse grained mapping FTL algorithms.

4.3 Summary of Measurement Results

According to our measurement results, SSD A and B satisfy the performance requirements for VoD storage. These SSDs show high throughput for large random reads and large sequential writes, and the random read throughput is also stable while sequential write operations happen simultaneously. Moreover, they are cheaper than the high-end SSD C, which is an important attribute for building a cost-effective VoD system.

This result is very interesting. Every SSD has its own FTL mapping algorithm, and with the mapping scheme, it hides the physical characteristics of NAND flash memory. NAND flash memory can be written only in large blocks, and thus, small and scattered updates are difficult to handle for an SSD. In general, higher class SSDs show better small random write performance by using more complicated FTL mapping algorithm and more resources. However, in VoD storage, workloads mainly consist of large sized requests, and small random write performance is not very important. Rather, complicated mapping schemes may cause unexpected delays as shown in SSD C due to background garbage collections and wear-leveling operations.

From these results, we can enumerate the flash memory SSD requirements for VoD storage.

1. It should have low cost per gigabyte to be cost-effective.

2. The random read throughput should be able to reach its maximum throughput with a sufficiently large request size.

3. The sequential write operations should not seriously affect the random read performance, and the deviation of the random read throughput should be as small as possible.

4. The garbage collection background process in the FTL layer should be controllable or predictable for the real-time requirement of video service.

5. HOW TO UTILIZE FLASH SSD

Given the measurement results, we now discuss how best to exploit flash-based SSD in a VoD storage. Even though the cost per gigabyte of flash SSD is decreasing, it is expected that flash SSD will have difficulty competing with magnetic disks in terms of capacity. Considering that only a small portion of popular files are requested frequently in VoD services, we claim that buffer cache is the best way to utilize the flash memory SSD, and the magnetic disk is still best for permanent video storage.

Interval caching [8, 9] is a state-of-the-art caching algorithm using RAM as a buffer cache to serve a larger number of video streams for a VoD server. Interval caching exploits both the characteristic of RAM and the characteristic of the VoD access pattern very well, which are symmetric read/write performance of RAM and the short average interval length between requests for popular videos. With these features, interval caching optimizes RAM capacity utilization, and it is more cost-effective than magnetic disks in serving popular videos. On the other hand, we have learned the characteristics of flash memory from the measurement of various flash memory SSDs in Section 4. The prominent feature from the measurement is the asymmetric read/write performance of flash memory SSD. In all SSDs, the write performance is far worse than the read performance. Worse yet is the unpredictable nature of the garbage collection (GC) activity that runs in the background after a number of write operations. The GC activity degrades not only the write performance but also the read performance. In addition, flash memory has a limitation in the number of erase operations, which is a lifetime problem. Suppose that a flash memory SSD is being written with W MB/s, the capacity of the SSD is C GB, and the maximum erase operations of the NAND flash memory cell is N (e.g., N is 10000 for MLC NAND flash memory and 100000 for SLC NAND flash memory). Assuming the erase operations can be distributed perfectly evenly over the cells, the life time of the SSD device is

$$T = \frac{C \times 1024 \times N}{W \times 3600} \text{ hours}$$

For example, if we write data onto a MLC flash memory SSD with 100MB/s and the capacity of the SSD is 128GB, the life time of the SSD is 3641 hours (152 days). Interval caching keeps writing data onto a device to serve the "following" streams, therefore, the flash memory SSD will quickly wear out when we use interval caching with it. Characteristics of flash memory SSD such as asymmetric performance and limited number of erase operations do not match the feature of the interval caching that is continuous write operations.

Traditional caching algorithms employed by various systems (e.g., database buffer manager, operating system memory manager) are LRU or CLOCK [10, 23] that operate at the block-level. These block-level caching algorithms, however, do not make sense for a VoD server for the following reasons. First, the block-level caching algorithm like LRU does not guarantee cache hit. It can impose unexpected load onto disks at inopportune times exactly when cache misses are being serviced from the disk. Second, a VoD server should guarantee continuous delivery of a video stream, but it is difficult for the server to guarantee continuous delivery when only some blocks of a video are in the cache buffer. For these reasons, we claim that a file-level caching algorithm should be used with flash memory SSD because SSD has sufficient capacity and can support continuous delivery of streams by caching entire video files. Because popular video files are accessed frequently and it is not trivial to define what is *least recent* for the file-level caching, we propose to use file-level Least Frequently Used (LFU) caching algorithm with flash memory SSD.

Interval caching cannot be used when the VoD server supports fast-rewind or fast-forward functionalities because the interval caching scheme assumes that the viewing streams for the same file are progressing at the same data rate. On the other hand, file-level LFU can support those functionalities because it stores a whole file in the buffer enjoying the cost-effective large capacity of flash memory.

Algorithm 1 Admission control for interval caching

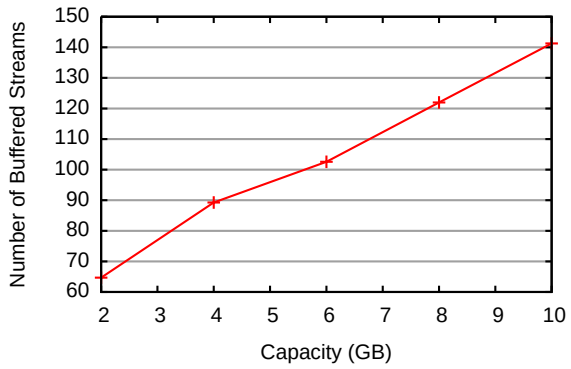
```
X ⇐ Disk bandwidth used by existing streams
Y ⇐ Bandwidth required by a new video request
Z ⇐ Maximum disk bandwidth
if X+Y > Z then
  {Disk bandwidth is the only criterion for admission.
  It's because a new stream should read data from disk
  initially even though the stream can be served from
  buffer later by interval caching.}
  REJECT the request
else
  ADMIT the request
end if
```

Algorithm 2 Admission control for file-level LFU

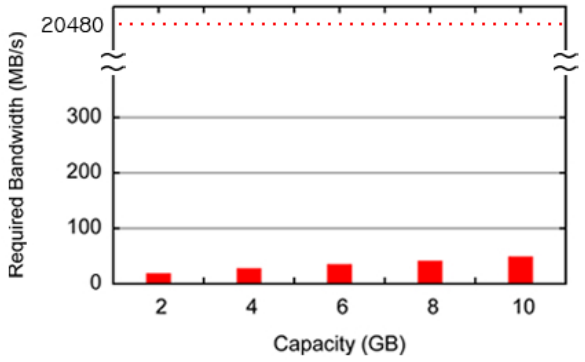
```
A ⇐ SSD bandwidth used by existing streams
B ⇐ Maximum SSD bandwidth
X ⇐ Disk bandwidth used by existing streams
Y ⇐ Bandwidth required by a new video request
Z ⇐ Maximum disk bandwidth
if A video request hits SSD cache then
  if A+Y ≤ B then
    ADMIT the request
  end if
end if
if X+Y ≤ Z then
  ADMIT the request
else
  REJECT the request
end if
```

6. EVALUATION

In this section, we will evaluate the performance of two devices, RAM and flash memory, via simulation when they are utilized as a buffer cache in a VoD server. Interval caching is applied for RAM, and file-level LFU is used for flash memory. We define the number of buffered streams as the num-

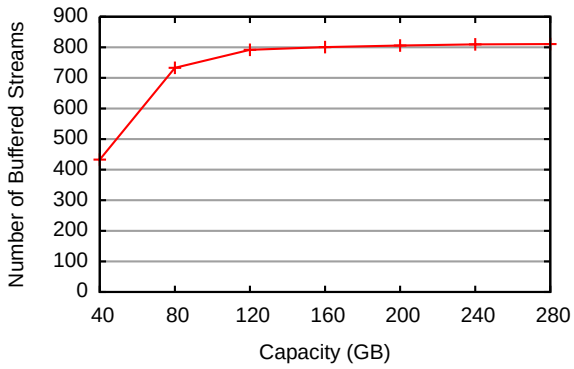


(a) Number of buffered streams increases proportional to the capacity of RAM

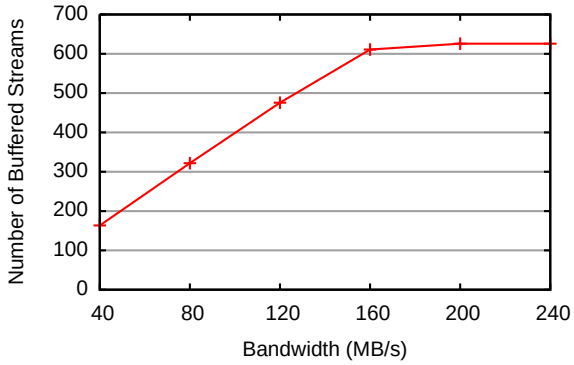


(b) Large unused bandwidth of RAM

Figure 9: Interval Caching with RAM



(a) Bandwidth is constant at 200MB/s. Number of buffered streams reaches a plateau due to the bandwidth limitation of flash memory.



(b) Capacity is constant at 64GB. When the bandwidth is large and the capacity is small, small number of video streams are served by flash memory because of small number of videos cached in flash memory. Then, it cannot utilize all the available bandwidth.

Figure 10: File-level LFU with flash memory

ber of streams served from a buffer cache (e.g., RAM or flash memory). Magnetic disks are used as permanent video storage.

6.1 Simulation

We assume all video files have the same and constant bitrate, 245KB/s, which is a typical bitrate for a high-quality video streamed over Internet. All video files have the same length, 60mins, if not mentioned otherwise. Therefore, the size of a video file is 862MB when the length of the video is 60mins. We assume 300 distinct videos are stored on the disks. The capacity and the bandwidth of the disks are 584GB and 100MB/s, respectively. The bandwidth comes from random read throughput measurement with 1MB I/O request size of 4 HDDs striped by RAID0 of our storage server. The HDD is SEAGATE Cheetah 15K.6 146GB and the RAID controller is LSI 1068E. The buffer for interval caching is modeled as a collection of 1MB blocks. We model the user request arrival as a Poisson process and model the video popularity as a Zipf distribution [8, 26]. We assume that the most popular N video files are already staged in the flash memory for ease of implementation.

Control parameters are divided into two sets, device parameters and workload parameters. Device parameters are the capacity and the bandwidth of each device, i.e., RAM and flash memory. Workload parameters include user request arrival rate, the video popularity distribution, and the video length. Performance metrics are average number of buffered streams, rejection probability, and hit ratio. The average number of buffered streams is defined as the number of streams that is served by the buffer cache (i.e., RAM or flash memory) averaged over total simulation duration. Hit ratio is defined as the number of reads served from the buffer cache divided by the total number of reads served by disks and the buffer cache. Rejection probability is defined as the number of rejected requests divided by the total number of requests that arrive at a VoD server. A VoD server must employ admission control algorithms to determine whether a new stream can be serviced without affecting streams already being serviced. Algorithm 1 shows the admission control for interval caching, and Algorithm 2 is that for file-level LFU. The duration of simulation is 10 hours.

6.2 Device Parameters

In this section, we investigate how bandwidth and capacity of a device affect the performance of the caching algorithm used with the device. The performance metric is the average number of buffered streams in this experiment. The Zipf distribution parameter is 0.271, and the arrival rate of the Poisson process is 0.5, which translates to 1 request every 2 seconds on an average.

Figure 9(a) shows the number of buffered streams using in-

terval caching with different RAM capacity, and Figure 9(b) is the required bandwidth to serve the streams at each capacity. For example, when the RAM capacity is 4GB, the interval caching can service 89 streams, and the required bandwidth for that number of streams is 21MB/s. We can see the linear relationship between the number of streams supported by the interval caching and the RAM capacity from Figure 9(a). The available bandwidth of RAM (20GB/s) is plotted as the dashed horizontal line in Figure 9(b). From the figure, We can notice that most of the available bandwidth of RAM is not utilized. This result tells us the scalability of interval caching with RAM is limited by the capacity of RAM, and the bandwidth of RAM is not a limiting factor.

Figure 10(a) shows the number of buffered streams by file-level LFU with different flash memory capacity. In this experiment, the bandwidth of the flash memory is set at 200MB/s. Different from Figure 9(a), the number of streams serviced by the buffer cache reaches a plateau beyond a capacity of 120 GB. At that point, 800 streams are served from the buffer cache and the cumulative bandwidth requirement for these streams from the flash-based SSD is 190MB/s. This means that while the flash memory could store more video files with a larger capacity, there is insufficient bandwidth to accommodate the real-time streaming requirements of the requests. On the other hand, Figure 10(b) demonstrates the number of buffered streams served by file-level LFU with different flash memory bandwidth. In this experiment, the capacity of the flash memory is set at 64GB. Similar to Figure 10(a), the number of buffered streams gets saturated beyond bandwidth of 160MB/s. When the bandwidth is large and the capacity is small, only a limited number of streams can be served by flash memory due to space limitation. In this case, the buffer cache is not able to fully utilize the available bandwidth.

From these results, we can learn the following: First, we do not need to worry about the bandwidth when we use interval caching with RAM. The only concern is the capacity, and we can scale up the number of streams with more capacity. On the other hand, we should worry about both the capacity and the bandwidth when we use file-level LFU with flash memory to scale up the number of streams. Flash memory SSD designer should design the SSD architecture to increase the capacity and the bandwidth both by maximizing the parallelism of flash memory chips to make a flash memory SSD suitable for VoD storage. Agrawal et al. have proposed a couple of possible architectures to achieve maximum bandwidth of the flash memory SSD such as parallel request queuing, individual data path to each flash memory chip, or interleaving [6].

6.3 Workload Parameters

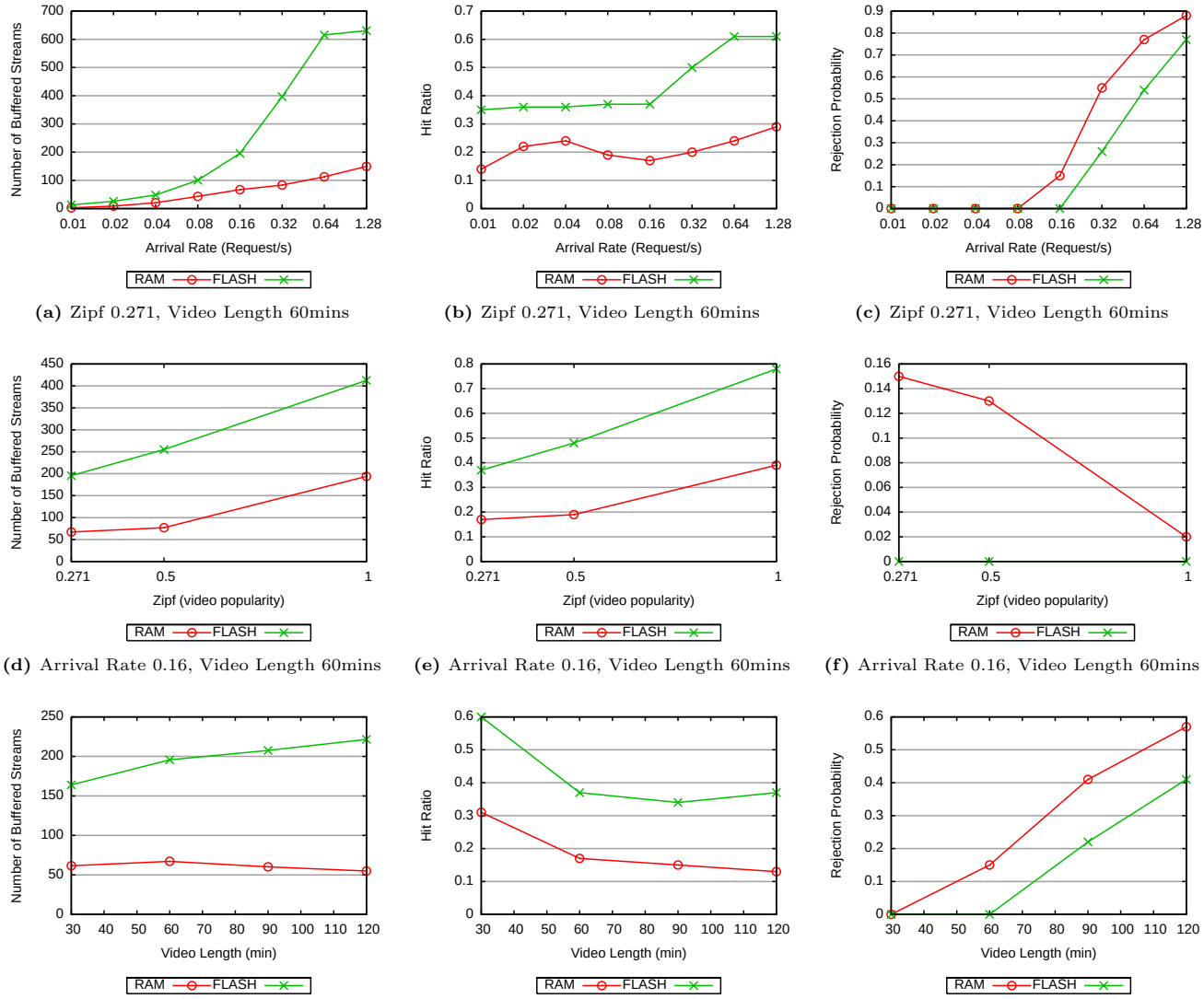
In this section, we investigate how the workload parameters affect the performance of interval caching with RAM and file-level LFU with flash memory. We fix the device parameters as follows. The capacity of RAM is 5.23GB, the capacity of flash memory is 64GB, and the bandwidth of flash memory is 155MB/s. We assume the bandwidth of RAM is infinite because we have learned that it is not a limiting factor from Section 6.2. We use 5.23GB RAM and 64GB flash memory because they have similar cost according to the cost per gigabyte of the devices in Table 1. The bandwidth for the flash memory comes from the measurement of SSD A from Section 4.

Figure 11(a) shows that the number of buffered streams increases proportional to the arrival rate for both interval caching with RAM and file-level LFU with flash memory. With a faster arrival rate, more requests can arrive to the VoD server within the same amount of time, and it makes the inter-arrival time between streams shorter. Therefore, interval caching can serve more streams with a faster arrival rate with a given RAM capacity. On the other hand, we can see that the flash memory cannot serve more than 647 streams limited by the bandwidth (i.e., 155MB/s). Note that 155MB/s bandwidth can accommodate at most 647 video streams when the video bitrate is 245KB/s. For similar reasons, the hit ratio of interval caching increases with a faster arrival rate, and the hit ratio of file-level LFU increases but saturates beyond 0.64 in which flash memory uses its full bandwidth. Figure 11(b) demonstrates this. Figure 11(c) shows the rejection probability as a function of the arrival rate. The rejection probability is 0 initially and then rises approximately linearly. When the arrival rate gets faster, it increases not only the arrival rate of popular videos but also that of unpopular videos. The requests for the unpopular videos that are not served by a buffer cache will go to disks, and they are rejected when the disks bandwidth is fully used. Considering the arrival rates where the rejection probability become non-zero (0.08 for interval caching and 0.16 for file-level LFU), we can see that the effect of interval caching is to increase the system capacity by 23% (hit ratio 0.19), and file-level LFU increases the system capacity by 59% (hit ratio 0.37).

With file-level LFU, 200 streams are served from the flash memory when the arrival rate is 0.16, consuming only 31% (200/647) of the available bandwidth of the flash memory. Therefore, when the arrival rate gets faster than 0.16, the increasing rejection probability is due to the disk bandwidth limitation (i.e., 100MB/s) because the flash memory could have served more streams with unused bandwidth. From this we can learn that much of the flash memory bandwidth would not be utilized if the disk bandwidth is too small.

Figures 11(d), 11(e), 11(f) show the performance of both caching schemes gets better with the increasing Zipf parameter. Larger Zipf parameter means more skewed video popularity. Therefore, the larger Zipf parameter makes the inter-arrival time shorter between streams for popular video files. Then, interval caching can accommodate more streams in the given RAM capacity, and file-level LFU also can serve more streams with the given flash memory bandwidth. This is why the rejection probability of interval caching could be greatly improved with larger Zipf parameters.

Figures 11(g), 11(h), 11(i) show the performance of both schemes as a function of video length. The number of buffered streams of interval caching is independent of the length of a video. It only depends on the length of intervals between streams when the bitrate and the buffer capacity are fixed. However, the buffered streams will hold the buffer space in the interval cache for a longer duration when the video length increases. Therefore, requests that cannot be served by interval cache will go to disks and they can be rejected when the disks bandwidth is fully used. That is why the hit ratio decreases and the rejection probability increases with longer video length for interval cache (Figures 11(h) and 11(i)). For file-level LFU, when the length of video in-



(a) Zipf 0.271, Video Length 60mins **(b)** Zipf 0.271, Video Length 60mins **(c)** Zipf 0.271, Video Length 60mins

(d) Arrival Rate 0.16, Video Length 60mins **(e)** Arrival Rate 0.16, Video Length 60mins **(f)** Arrival Rate 0.16, Video Length 60mins

(g) Zipf 0.271, Arrival Rate 0.16 **(h)** Zipf 0.271, Arrival Rate 0.16 **(i)** Zipf 0.271, Arrival Rate 0.16

Figure 11: Performance comparison of Interval Caching with RAM and File-level LFU with Flash memory

creases, less video files can be stored in the flash memory with a given capacity. Therefore, less video requests hit the buffer cache for a given arrival rate. We can see decreasing hit ratio and increasing rejection probability beyond a video length of 60 mins in Figures 11(h) and 11(i) for file-level LFU.

Note that all streams served by the flash memory can deliver video blocks without hiccups because a whole video file is cached in the flash memory buffer and the VoD server guarantees the streaming rate to be the video bitrate by the admission control mechanism.

6.4 Cost-Effectiveness

In this section, we compare the cost-effectiveness of three different storage configurations. First configuration is HDD only. Only hard disk drives are used for storing video files and serving user requests. Second configuration is DRAM with HDD. DRAM is used for a buffer cache, and the interval caching is applied to it. HDDs are used for permanent storage. Third configuration is flash memory SSD with HDD. The SSDs are used for a buffer cache, and the file-level LFU is applied to it. HDDs are used for permanent storage.

The parameters used for the analysis is listed in Table 3: 60 mins video length, 1 request/sec arrival rate, and the 0% rejection probability translates to a peak requirement of 3600 concurrent video streams to be supported by the VoD server. We use a following model for calculating the maximum number of streams that can be served by HDDs or flash memory SSDs.

$$S = \frac{T}{B}, T = R \times N$$

where S is maximum number of video streams, T is total read throughput of devices, B is video bitrate, R is random read throughput of a device, and N is the number of devices.

The number of streams served by DRAM cannot be calculated simply as above, since it is determined by the peak number of concurrent streams that interval caching can serve

Parameters	Value
Video Library	300
Zipf (Video Popularity)	0.271
Video Bitrate	245 KB/s
Video Length	60 mins
Arrival Rate	1 request/sec
Rejection Probability	0 %

Table 3: Simulation parameters used for cost-effectiveness analysis.

Device	Capacity (GB)	Random Read Throughput (MB/s)	Cost ($)
DRAM	1	∞	23
HDD15K	146	70	180
MLC SSD	64	155	120

Table 4: Specification of different devices. DDR3 SDRAM is used as a representative of DRAM. We assume the random read throughput of the SDRAM is infinite. SEAGATE Cheetah 15K.6 is used for HDD15K, and RiData NSSD-S25-64-C06MPN is used for MLC SSD. The random read throughput of HDD15K and MLC SSD is measured by *xdd* benchmark program when I/O request size is 1MB.

for a given DRAM capacity for the duration of simulation, which is 10 hours. We assume the bandwidth of DRAM is infinite because the maximum number of streams that can be served by the interval caching is bounded by capacity of the DRAM in most cases.

Using the simulator mentioned in Section 6.1 and the specification of each device listed in Table 4, we calculate the combination of devices for each storage configuration that needs minimum cost and meets the workload requirements of Table 3. From the simulation results, we can see that 13 HDDs are needed for the configuration of HDD only, and its cost is 1.46 streams per dollar. For DRAM with HDD, 2 DRAM modules and 12 HDDs are needed, and its cost is 1.54 streams per dollar. For SSD with HDD, 5 SSDs and 2 HDDs are needed, and its cost is 3.55 streams per dollar. Overall, the storage configuration of SSD with HDD is the most cost-effective solution which meets the workload requirements. It can serve two times more streams than HDD only or DRAM with HDD for the same dollar investment.

6.5 Evaluation Results Summary

From the results, we have learned the following:

1. We need to consider the capacity and the bandwidth when we choose a flash memory SSD for VoD storage. Moreover, flash memory SSD designer should care about the architecture of the device to increase both the capacity and the bandwidth.

2. When we use flash memory SSD with HDD, the capacity and the bandwidth of flash memory should be in balance with those of disks to fully utilize the capacity and the bandwidth benefits of flash memory with a given QoS criterion (e.g., rejection probability),

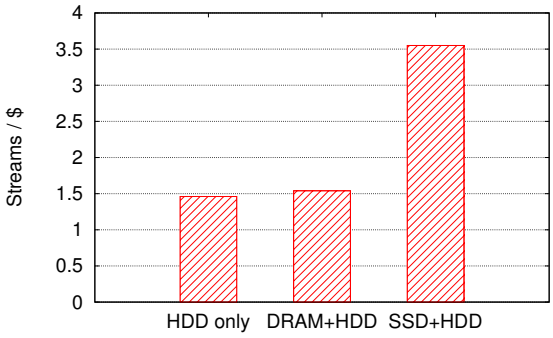


Figure 12: With a given cost, DRAM with HDD can serve slightly more streams than HDD only, but SSD with HDD can support two times more streams than DRAM with HDD.

3. Flash memory SSD with HDD is the most cost-effective solution compared to DRAM with HDD or HDD only.

7. RELATED WORK

Research adopting flash memory to a web server and a database server has been done. Kgil et al. [17] have studied energy efficient web server using flash memory as an extended system memory. By their simulation experiments, using NAND flash incorporated architecture has improved performance by 11% and saved power by 75% for a web server. Lee et al. [19, 18] have researched the application of flash memory SSD to a database server. The authors claim that a single enterprise class SSD can be on a par with or far better than a dozen spindles with respect to transaction throughput, cost effectiveness and energy consumption. Despite the successful research regarding the applicability and effectiveness of flash memory for a web server [17] and a database server [18], flash memory adoption to a VoD server has not been studied yet.

Narayanan et al. [22] have analyzed the efficacy of using Flash-based SSD for enterprise class storage via simulation. In addition to using Flash exclusively as the permanent storage, they also study a tiered model wherein the SSD is in between the RAM and the disks. In their study they use enterprise class SSD ($23/GB). Their conclusion is that SSD is not cost-effective for most of the workloads they studied unless the cost per gigabyte for SSD drops by a factor of 3-3000. Their results are predicated on the assumption that SSD is used as a transparent block-level device with no change to the software stack (i.e., application, file system, or storage system layers). The results we report in this paper offers an interesting counter-point to their conclusion. In particular, we show that the use of inexpensive SSDs as a buffer cache is a cost-effective alternative for structuring a VoD server as opposed to increasing either the disk bandwidth or RAM capacity to meet a given QoS constraint.

8. CONCLUSIONS

With the increasing demand for high bitrate video, there is a need to rethink the storage architecture of a VoD server. It is not cost-effective to simply rely on disk and DRAM alone to scale up the VoD server to meet this increased demand. Disk as the permanent store for video, offers the

much needed capacity to store large number of videos. However, a pure disk-based VoD quickly becomes bandwidth limited and cannot meet the latency requirements for real-time playback of video as the workload scales up without a significant investment on the disk infrastructure. DRAM as a buffer cache offers the necessary low latency for recently accessed video and has been deployed to address the bandwidth problem in a disk-based VoD. However, DRAM technology, due to the prohibitive cost, is feasible only for a modest sized buffer cache. Flash-based SSD is an attractive alternative to consider as a buffer cache instead of DRAM since it has much better speed of access compared to a disk, and offers much more capacity for the same dollar investment compared to a DRAM. We have explored this alternative by conducting extensive studies in this paper. The studies reveal very interesting insights. The first non-intuitive result was the revelation that low-end SSDs with simple FTLs are better suited for use as a buffer cache in a VoD server given the VoD workload characteristics. Second, we determined that SSD using file-level LFU provides comparable and at times superior performance to DRAM as a buffer cache. Third, the cost of engineering a VoD server with flash-based SSD as a buffer cache and HDD as permanent storage will be significantly cheaper than DRAM with HDD or HDD only.

9. REFERENCES

[1] Enterprise ssds. `http://www.stec-inc.com/downloads/whitepapers/Benchmarking_Enterprise_SSDs.pdf`.

[2] Hulu. `http://www.hulu.com`.

[3] Netflix. `http://www.netflix.com`.

[4] Xdd. `http://www.ioperformance.com`.

[5] Youtube. `http://www.youtube.com`.

[6] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. Manasse, and R. Panigrahy. Design tradeoffs for ssd performance. In *ATC'08: USENIX 2008 Annual Technical Conference on Annual Technical Conference*, pages 57–70, Berkeley, CA, USA, 2008. USENIX Association.

[7] F. Chen, D. A. Koufaty, and X. Zhang. Understanding intrinsic characteristics and system implications of flash memory based solid state drives. In *SIGMETRICS '09: Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*, pages 181–192, New York, NY, USA, 2009. ACM.

[8] A. Dan and D. Sitaram. Buffer management policy for an on-demand video server. Ibm research report rc19347, T.J. Watson Research Center, Yorktown Heights, NY, USA, 1994.

[9] A. Dan and D. Sitaram. A generalized interval caching policy for mixed interactive and long video environments. In *Proceedings of Multimedia Computing and Networking Conference*, San Jose, CA, USA, 1996.

[10] A. Dan and D. Towsley. An approximate analysis of the lru and fifo buffer replacement schemes. In *Proceedings of the ACM SIGMETRICS*, pages 143–152, Denver, CO, USA, May 1990.

[11] J. Gemmell, H. M. Vin, D. D. Kandlur, P. V. Rangan, and L. A. Rowe. Multimedia storage servers: A tutorial. *Computer*, 28(5):40–49, 1995.

[12] G. Graefe. Integrating flash devices. *Communications of the ACM*, 52(4):97–97, April 2009.

[13] J. Gray and B. Fitzgerald. Flash disk opportunity for server-applications. *http://www.research.microsoft.com/~gray*, January 2007.

[14] C.-G. Hwang. Nanotechnology enables a new memory growth model. In *Proceedings of the IEEE 91(11)*, pages 1765–1771, November 2003.

[15] Intel Corporation. Understanding the Flash Translation Layer (FTL) Specification. White Paper, `http://www.embeddedfreebsd.org/Documents/Intel-FTL.pdf`, 1998.

[16] A. Kawaguchi, S. Nishioka, and H. Motoda. A flash-memory based file system. In *USENIX Winter*, pages 155–164, 1995.

[17] T. Kgil, D. Roberts, and T. Mudge. Improving nand flash based disk caches. In *Proceedings of the 35th International Symposium on Computer Architecture*, pages 327–338, June 2008.

[18] S.-W. Lee, B. Moon, and C. Park. Advances in flash memory ssd technology for enterprise database applications. In *Proceedings of the ACM SIGMOD*, pages 863–870, June 2009.

[19] S.-W. Lee, B. Moon, C. Park, J.-M. Kim, and S.-W. Kim. A case for flash memory ssd in enterprise database applications. In *Proceedings of the ACM SIGMOD*, pages 1075–1086, June 2008.

[20] A. Leventhal. Flash storage memory. *Communications of the ACM*, 51(7):47–51, July 2008.

[21] M-Systems. Two Technologies Compared: NOR vs. NAND. White Paper, `http://www.dataio.com/pdf/NAND/MSystems/MSystems\_NOR\_vs\_NAND.pdf`, 2003.

[22] D. Narayanan, E. Thereska, A. Donnelly, S. Elnikety, and A. Rowstron. Migrating server storage to ssds: Analysis of tradeoffs. In *Proceedings of the ACM EuroSys*, Nuremberg, Germany, April 2009.

[23] V. F. Nicola, A. Dan, and D. M. Dias. Analysis of the generalized clock buffer replacement scheme for database transaction processing. In *Proceedings of the ACM SIGMETRICS*, pages 35–46, 1992.

[24] C. Park, W. Cheon, J. Kang, K. Roh, W. Cho, and J.-S. Kim. A reconfigurable ftl (flash translation layer) architecture for nand flash-based applications. *Trans. on Embedded Computing Sys.*, 7(4):1–23, 2008.

[25] A. R. Rahiman and P. Sumari. Solid state disk: A new storage device for video storage server. In *Proceedings of the International Symposium on Information Technology*, pages 1–8, August 2008.

[26] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng. Understanding user behavior in large-scale video-on-demand systems. In *Proceedings of the ACM EuroSys*, April 2006.