Achieving Reliable Parallel Performance in a VoD Storage Server Using Randomization and Replication [*]

Yung Ryn Choe and Vijay S. Pai
Purdue University
West Lafayette, IN 47907
{yung, vpai}@purdue.edu

Abstract

This paper investigates randomization and replication as strategies to achieve reliable performance in disk arrays targeted for video-on-demand (VoD) workloads. A disk array can provide high aggregate throughput, but only if the server can effectively balance the load on the disks. Such load balance is complicated by two key factors: workload hotspots caused by differences in popularity among media streams, and "fail-stutter" faults that arise when the performance of one or more devices drops below expectations due to manufacturing variations, hardware problems, or geometry-related variations.

This paper focuses on the random duplicate assignment (RDA) data allocation policy which places each data block on two disks chosen at random, independent of other blocks in the same media stream or other streams. This strategy is compared to traditional single-disk file allocation, disk striping (RAID-0), disk mirroring (RAID-1), and randomization without duplication. The various allocation schemes are implemented and tested using a prototype VoD server with 2 dual-core Opteron processors, 8 SATA disks, and 4 Gigabit Ethernet interfaces running the Linux 2.6 kernel. The results indicate that combining randomization and replication allows RDA to effectively tolerate both workload hotspots and fail-stutter faults better than previous schemes.

1 Introduction

Recent years have seen a dramatic increase in computational capability and network bandwidth available to end-users in commercial, home, and mobile environments. The long-running trend of exponential performance growth per unit cost has now yielded systems ranging from ubiquitous handheld entertainment devices to high-end desktops connected through Gigabit Ethernet. All levels have seen demand for richer and more targeted content. For example, the mobile space has seen the recent arrival of the video iPod as a tool to enable on-demand playback of saved video clips such as broadcast TV shows, as well as offerings such as Verizon V-CAST which have brought video offerings to cell phones. High-end entertainment has seen a rapid increase in pay-per-view offerings and special-interest (e.g., regional) satellite channels. Even the educational environment has seen offerings such as iTunes U, which provides video and audio captures of classroom lectures over the Web. Such trends motivate investigations into video-on-demand (VoD) servers. Substantial previous work has analyzed, prototyped, and built VoD servers [5, 9, 14, 36]. However, more recent advances in storage technologies enable such servers to be built in a cost-effective and storage-efficient manner, requiring substantially different design strategies.

Rapid improvements in magnetic and electromechanical technologies have facilitated greater information densities and capacities in hard disk drives [13]. Commodity hard drive capacities have grown as much as 100% per year since 1997, now reaching capacities as great as 500 GB with prices under $1 per Gigabyte. The use of point-to-point links in modern Serial ATA (SATA) interfaces has also allowed for the construction of cost-effective and resource-efficient disk arrays, since individual disks no longer need to contend for shared bus resources as in the older Ultra ATA and SCSI standards. Such disk arrays are ideally-suited as building blocks for video-on-demand (VoD) servers. Since a modern 500 GB commodity disk can hold more than 100 DVD-quality (4.7 GB) multimedia streams, a single 2U video-on-demand server could hold 800 movies. A standard 42U rack of such servers could hold over 16000, dwarfing the corner video store. Five such racks would yield a greater selection than Netflix (65,000 movies as of 10/1/2006). Although modern systems can easily accommodate the capacity requirements of VoD storage, it is far more challenging for the server to actually schedule all the distinct request streams to the much smaller number of disks while meeting real-time delivery targets.

While capacity has been growing at an unrivaled clip, disk internal data rate (sequential access bandwidth from the media to the drive's own buffers) has been improving at about 40% per year; disk access latency has only been improving about 15% per year [13]. Thus, a high capacity disk can only achieve good performance if its workload consists primarily of large sequential transfers. Although a disk array can achieve high aggregate throughput by employing parallel disks, such parallelism is only possible if the system achieves good load balance across the disks. Achieving such balance is complicated by workload hotspots and variations, as well as by "fail-stutter faults": performance degradations caused by manufacturing variations, by hard-

[*] This work is supported in part by the National Science Foundation under Grant Nos. CCF-0532448 and CCF-0621457.

1-4244-0910-1/07/$20.00 ©2007 IEEE.

ware glitches such as seek errors and bus timeouts, or by geometry-related variations (e.g., slower reads toward the center of the drive platter as a result of lower linear momentum).

One approach to achieving load balance in the presence of workload hotspots is disk-striping, popularly called RAID-0. Although disk striping can provide ideal load balance by parallelizing requests across disks, it may also suffer from poor performance. Reddy and Banerjee observed that disk striping thus increases the effective service time of a request, reducing aggregate throughput of a disk array [25]. In particular, the actual transfer rates of striped data accesses will be accelerated by the degree of striping, but the seeks must take place across all the disks, creating occupancy at the disk heads. Arpaci-Dusseau further showed that the performance of a RAID-0 array degrades to the performance of the slowest disk, crippling performance in the case of a single-disk performance degradation [2]. Although designed primarily for redundancy and availability, RAID-1 (disk mirroring) may also improve performance by splitting a read-dominated workload between two disks [24].

Randomization is a useful alternative to disk striping to reduce the likelihood of hotspotting. In this strategy, data blocks are mapped to disks randomly, with any given block of data equally likely to reside on any disk. Hotspotting is thus eliminated statistically, and this strategy has been adopted in video-on-demand systems like RIO [29]. Theoretical results have nevertheless shown that randomization may also suffer from hotspotting. In particular, when D blocks are randomly selected from D disks, it can be expected with high probability that one disk has requests for $\Theta(\frac{\log n}{\log \log n})$ blocks, using balls-and-bins analysis [12]. Even with careful randomization, there are known to be problems such as external sorting which are likely to perform poorly [4].

Sanders et al. gave an elegant technique to avoid hotspotting by placing each logical data block on two randomly selected disks so that at least one copy can be read without I/O bottlenecks [28]. This strategy, called *random duplicate allocation* (RDA) trades off some capacity for more reliable performance, but the current rate of capacity growth may make this approach a reasonable tradeoff.

This paper explores the performance of various data allocation policies when applied to a disk array for serving DVD-quality video-on-demand streams. The policies of naive single-disk allocation, disk striping, disk mirroring, randomization, and random duplicate allocation are implemented on an actual prototype VoD storage server, using application-controlled mapping of blocks to disks. These strategies are tested under uniform workload distribution, various workload hotspot models, and single-disk fail-stutter fault models caused by degraded throughput from a disk. The results show that random duplicate allocation is the only strategy studied that performs well under each of the workload hotspotting models. RDA also suffers fewer and shorter client starvation incidents than most of the other schemes when exposed to single-disk fail-stutter faults. These results from an actual prototype storage server complement and experimentally confirm the prior theoretical analysis showing the advantages of RDA [28].

The remainder of this paper proceeds as follows. Section 2 gives relevant background information on video-on-demand servers and fail-stutter faults. Section 3 describes strategies for achieving reliable disk array performance. Section 4 describes the prototype server hardware and software, while Section 5 covers the experimental methodology. Section 6 presents the experimental results. Section 7 describes related work, and Section 8 concludes the paper.

2 Background

The network server field in general has achieved great performance improvements in recent years, with such strategies as event-driven software architectures and zero-copy I/O [15, 20, 22, 23, 34]. However, these works have focused primarily on workloads with small files and high disk cache hit ratios (e.g., web servers). The VoD workload is different, as each stream may be multiple Gigabytes and may by itself overwhelm the disk cache. The web proxy server workload sees substantial disk access, and some software strategies have focused on minimizing seeks [17]. However, those file sizes are still small compared to the VoD workload, and the performance levels reported by proxy benchmarks are less than 300 Mbps [27].

The key issue in the performance of video-on-demand (VoD) storage servers is managing the disk array. Hard disk drive capacity has been improving at a 100% annual rate since the introduction of GMR heads in 1997 [13]. At the same time, the internal data transfer rate of disks, measured for sequential access between the magnetic media and the driver's internal buffers has been improving at roughly 40% per year. These improvements have come about primarily through the greater information density on disk. Disk access latency has also improved, but only at about 15% per year. While the capacity improvements in disks certainly enable the storage of an ever-greater number of high-bitrate streams, the performance trends only support substantial improvements if the workload offered to the disks consists primarily of large sequential transfers. Storage servers achieve high data transfer rates by employing parallel disks, but effectively utilizing parallel disks for large data streams is challenging because of the need to multiplex a large number of distinct request streams onto a smaller number of disks while meeting real-time delivery targets. Additionally, the server must carefully balance the load across the disks to extract the maximum possible parallelism from the disk array.

There are two important ways in which disk arrays can fail to achieve their desired parallelism. One way is due to realistic workloads in which some media objects are far more popular than others. Chesire et al. studied client-based streaming-media workloads and found that the distribution of client requests to objects is Zipf-like with a Zipf parameter of 0.47 [6]. In other words, the number of requests to the object ranked r is roughly proportional to $r^{-0.47}$. Because the workload is Zipf-like, there will be files that are requested more frequently than others, causing hotspotting.

If files are not carefully spread across the disks, a workload hotspot will directly lead to disk load imbalance.

A second way for disk arrays to achieve less-than-expected parallelism is through the unexpected underperformance of a component. For example, Remzi and Andrea Arpaci-Dusseau cite several documented cases of disks underperforming when describing fail-stutter faults [3]. In one case, most disks delivered 5.5 MB/s sequential transfer bandwidth but one disk delivered only 5.0 MB/s due to bad block faults that were being remapped elsewhere on disk, introducing seeks. Van Meter pointed out that performance can vary by a factor of 2 across the different zones of a single disk; consequently, a disk reading from a slow zone will underperform another one in the same array that happens to be reading from a faster zone [19]. Other performance degradations occur for reasons that cannot be easily or directly analyzed.

3 Achieving Reliable Performance

Reliable performance in a parallel I/O system requires the system to support a high level of concurrency even in the presence of workload hotspots and hardware performance variations. Effective mapping of request data blocks to disks can be critically important in determining performance. Temporal variations in the workload and unpredictable hardware performance degradations make it impossible to find an ideal solution for all situations. However, some basic principles can help:

1. The entire contents of a stream should not be on one disk. Otherwise, severe disk load imbalances could arise for workloads with requests for multiple streams that happen to use only the same disk.
2. The granularity of allocating data contiguously should be at least as large as a single time slice in the stream. Otherwise, a single time slice from a single stream could require service from multiple disks, activating simultaneous seeks on each of those disks and thus reducing the aggregate throughput of the system. The remainder of this paper refers to such allocation units as *blocks*.
3. The performance of the system should not be dramatically impacted by the underperformance of a single disk.

Traditional file allocation in an operating system meets none of the above goals: a file is usually on a single disk, allocated with data blocks of 4 kB (compared to over 4 MB of data in a 5-second time slice of a DVD-quality video), and an underperformance on a single disk would impact any attempt by the system to access files on that disk. Modern extent-based file systems typically meet goal 2 above since they aim to put successive writes to a file in contiguous blocks on disk [18]. This paper refers to schemes that allocate each stream entirely on one disk as *naive*.

Disk striping (RAID-0) meets goal 1 by spreading data across all disks and may meet goal 2 by choosing the stripe size appropriately [11, 16, 31]. However, goal 3 is not met since prior work has shown that RAID-0 array performance tracks the slowest disk [2]. Disk mirroring (RAID-1) meets goal 2. Additionally, it may meet goal 3 by splitting the workload between the two disks in the mirror-pair intelligently [24]. However, RAID-1 does not meet goal 1 since hotspotting is only spread across a mirror-pair; this particular mirror still sees more load than the other mirrors. The RIO system stores media blocks on randomly selected disks to support multiple access patterns and stream popularity variations [29]. Such randomization can enable goal 1 by distributing popular content across the disks; goal 2 can be met by choosing the granularity appropriately. However, randomization alone may actually exacerbate the problem of single-disk underperformance since every stream access will now touch the underperforming disk.

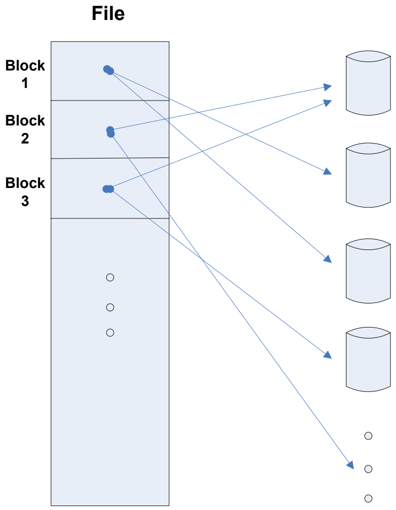

Figure 1. Depiction of assignment of data blocks to disks using the random duplicate assignment strategy.

To address limitations in randomization alone, the random duplicate allocation (RDA) strategy replicates every content block on two randomly-chosen disks [28]. Figure 1 illustrates how the blocks of a file are replicated and scattered across the disks under RDA. Although replication in RDA was proposed to help improve load balance and tolerate workload variations, it may also be useful to tolerate single-disk fail-stutter disk faults since the more properly functioning replica may be chosen preferentially over the degraded one. In general, replication requires resolving coherence issues on updates; this is not relevant here since the video-on-demand workload is write-once, read-many.

4 Prototype Implementation

Server hardware. This paper reports the performance of a prototype streaming storage server that employs modern multicore processors and a SATA disk array. The system specifically includes two 2.2 GHz dual-core AMD Opteron processors, four Gigabit Ethernet links, 4 GB of DRAM, two Promise FastTrak TX4200 4-port SATA RAID controller, and 8 Seagate SATA disks with 400 GB capacity and 7200 RPM rotation speed. The SATA controller is configured in JBOD (Just a Bunch Of Disks) mode for all tests. Performance tests using lmbench indicate that the disks see an average access time of 17 ms and a peak sequential transfer rate of 525 Mbps; there is negligible inter-disk contention since each disk has a 3 Gbps point-to-point link to the controller and the controller itself sits on an 8 Gbps PCI-X bus.

Server software. The software platform consists of a standard Linux 2.6 kernel. Because the operating system is unchanged, it may still allocate disk blocks at a minimum granularity of 4K; in practice, the filesystem tends to allocate file blocks sequentially. The disk array can hold over 650 4.7 GByte DVD-quality streams. However, these tests use an even larger number of 1 GByte streams to represent a broader selection of available content. Further, the tests only access the first 300 MB in order to allow the tests to complete more quickly. There is no substantial difference in performance between accesses to 300 MB and 4.7 GB streams; stream data is extremely unlikely to be reused in the filesystem cache in any case since each test overflows the cache in less than 30 seconds.

Stream blocks are mapped to the disk array using the allocation strategies discussed in Section 3: naive, RAID-0, RAID-1, random, and RDA. The mapping of stream blocks to disks is controlled through user-level code for all allocation strategies tested, allowing full flexibility in allocation without the complexities of operating-system coding. Test results show that user-level data management has less than 1% overhead compared to OS-level implementation of basic policies (such as naive file allocation). This is possible because no actual file data is passed to the user-level; the user-level only records disk numbers and offset positions for the stream data blocks. All interactions with the file data are performed using the Linux `sendfile` system call which directly dispatches data from a disk file to a network socket. The system also does not use hardware-controlled RAID because the controller does not support sufficiently large disk stripes and does not expose how it balances load in a disk mirror.

Figure 2 represents the flow of operations in the application-level server software. The server responds to a client connection by creating a new thread and then sending a media data block as an initial buffer. The media data block size is determined by the target bitrate and the time slice used for allocation. In this system, the target bitrate is 6 Mbps for DVD-quality streams and the time slice is 5 seconds. It then enters a repeated pattern of sending a data block, seeing how long it took to perform that transmission, and sleeping that connection for the remainder of the time slice represented by the block. If a transmission takes longer than that time slice, the server sends the next chunk at a higher rate (by sleeping for less time) until the client side catches up. Each time the transmission takes longer than the time slice, the server adds the excess time to a per-connection deficit variable and sets the target time for the next data block transmission be the standard time slice less the deficit. This deficit is reduced (or reset) if the next transmission completes before the target elapses. If the client side repeatedly fails to keep up, its buffer will be exhausted and it will starve; the server will continue to send at a higher rate until its target has been met. The prototype delivers the content via HTTP over TCP/IP using the application-level rate-pacing described above; realistic multimedia protocols would add some network overhead but would not change the disk access pattern.

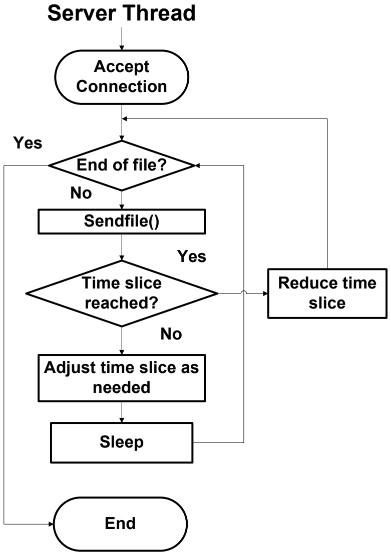

Figure 2. Flowchart describing operations in server thread for processing individual video-on-demand (VoD) client connection

5 Experimental Methodology

This paper evaluates the resiliency of various file allocation to workload hotspotting and fail-stutter faults. Each of these types of tests is conducted by measuring the performance of the prototype VoD storage server described in Section 4. Four client machines on the same LAN as the server initiate parallel requests for content from the server. The results report the maximum number of simultaneous connections that can be successfully supported by the server for the given workload. This maximum value is determined by binary search; any test in which any client connection starves even once after the initial buffering (namely, in which its data buffer empties because of slow server responses) is considered to have failed.

This paper explores three workload models. The first is uniform popularity, in which streams are chosen with equal likelihood from the collection of available streams and the workload is exactly balanced across the disks in the naive allocation scheme.

The second workload model uses Zipf-like request distributions, with a Zipf popularity parameter of 0.5 to represent workload hotspotting akin to that seen by Chesire et al. [6] and a Zipf parameter of 1.0 to model greater request hotspotting. Each of these tests also uses a Poisson request arrival model, with a mean interarrival time of 1 second. In each of these Zipf popularity models, the naive allocation strategy is split into "naive expected" and "naive worst". In the former, the files are allocated across the disks with no respect for their popularity ranking. In the latter, the files are allocated across the disks in such a way that the most popular files are clustered together on one disk, the second most popular set of files on the second disk, and so forth down the disk array. We do not consider "naive best" since the best possible placement of files on disks would depend greatly on ever-changing popularity and would thus require continual reshuffling.

For the third workload model, files known to be allocated on a specific disk or disk mirror are chosen with an extra likelihood in addition to a base uniform distribution; this distribution can be thought of as having between 20–80% overloading of one of the disks. This workload model only applies to naive and RAID-1 since RAID-0, random, and RDA all scatter accesses to all disks.

The paper also considers the impact of single-disk fail-stutter faults in which a disk achieves degraded sequential transfer rates, as in the examples discussed in Section 2. To model the impact of a single underperforming disk, each server access to that disk also invokes an additional transfer that reads in an amount of useless data proportional to the amount of requested data. For example, a 1 MB read combined with a 20% additional read will actually read 1.2 MB for every 1 MB of useful data. The VoD server is then subjected to a uniform request workload at the throughput level that was achieved by the performance tests. The results report the number of times that client connections starve for the given fault model, the average duration of each starvation incident, and the maximum duration of the starvation incidents seen for each allocation policy.

6 Experimental Results

6.1 Uniform Workload Distribution

Figure 3 shows the performance of the various stream allocation strategies when subjected to a uniform workload distribution. In this distribution, the clients request distinct streams that are chosen in such a way as to provide a completely balanced workload even with the naive allocation strategy. Under this workload, each of naive allocation, random duplicate allocation, and RAID-1 perform nearly identically, achieving 332–336 simultaneous 6 Mbps DVD-quality connections without ever starving the client connections. It is worth noting that these results, with an aggregate throughput of over 2 Gbps, exceed any previously reported numbers from single-machine academic or industrial VoD servers of which we are aware. (Specific performance citations are provided in Section 7.)

The random allocation strategy achieves 288 connections; randomization may actually underperform naive allocation in a uniform workload since the effects of randomness may actually create some load imbalances as some disks are randomly favored over others. RDA avoids this problem because it always chooses the less heavily-loaded replica for a given data block. RAID-0 performs the worst, at 256 connections. RAID-0 actually sees load imbalance for this workload because the connections start nearly simultaneously and all the streams start their disk striping on disk 0. Similar problems have been observed with disk striping in problems such as external sort [33].

6.2 Workload Hotspotting

Figure 4 presents tests where the clients choose from the set of streams randomly with a Zipf distribution using a popularity parameter of 0.5 and 1.0. A parameter of 0.5 is similar to the actual popularities seen by previous work analyzing multimedia workloads [6]. The performance of naive allocation degrades by about 10% in the expected case and by nearly 50% in the worst case. Recall that naive allocates each file on a single disk, with the popular files split among the different disks randomly in the expected case and clustered onto the same disks by order of popularity in the worst case. The other schemes perform similarly to, or slightly better than, their performance in the uniform distribution. Skewing may degrade performance by introducing load imbalance but may also help performance by allowing some file cache effectiveness; these effects seem to be roughly in balance for all but the naive allocations. The benefits of RDA and RAID-1 over the other schemes are slightly more pronounced here than in the other cases, with a 14% lead over the nearest competitor (naive expected).

A Zipf distribution parameter of 1.0 represents a still more skewed distribution. Here the positive effects of skewing in improving cache reuse are much more evident, as all but the naive worst-case allocation perform better than in the uniform distribution. RDA outperforms naive-worst by more than double, outperforms naive-expected by 5%, and outperforms RAID-1, Random, and RAID-0 by 15–20%.

Figure 5 examines another workload hotspotting model, where specific disks or disk mirrors are overloaded with a greater request rate. This hotspot model only applies to naive and RAID-1, since the other schemes already distribute files across all the disks. The number on the X-axis represents the overload percentage applied to the target disk or disk mirror above and beyond the base uniform distribution. For example, an overload of 40% indicates that the target disk will see 40% more load than the other disks. Note that the overload rates are not exactly 20, 40, 60, and 80% because the number of streams applied in overload must be an integer and may not divide nicely with the base value. The performance of the naive allocation strategy drops nearly 40% when an 82.5% overload is applied to one disk, while the RAID-1 performance drops 30% when a 75% overload is applied to one disk. This sort of overloading does not apply to RDA, random, or RAID-0, and thus does not affect their performance.

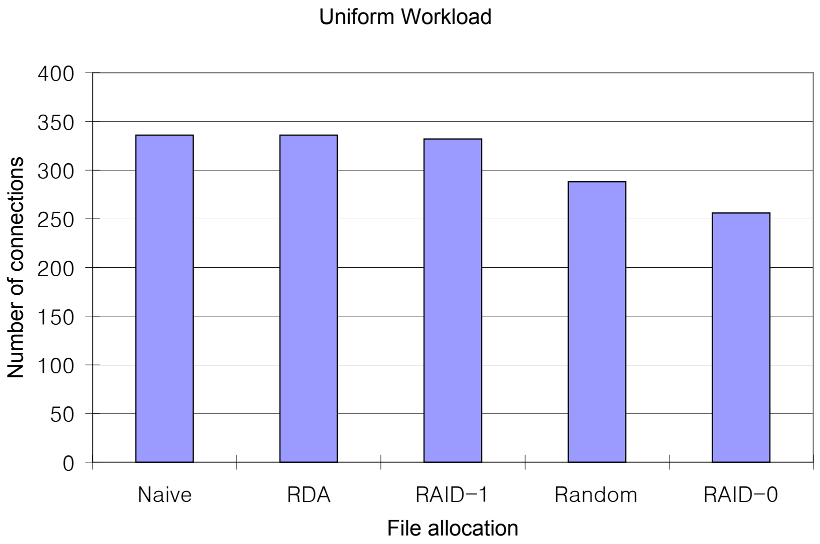

Figure 3. Performance under uniform workload measured in terms of number of simultaneous DVD-quality streams supported by storage server under various allocation policies.

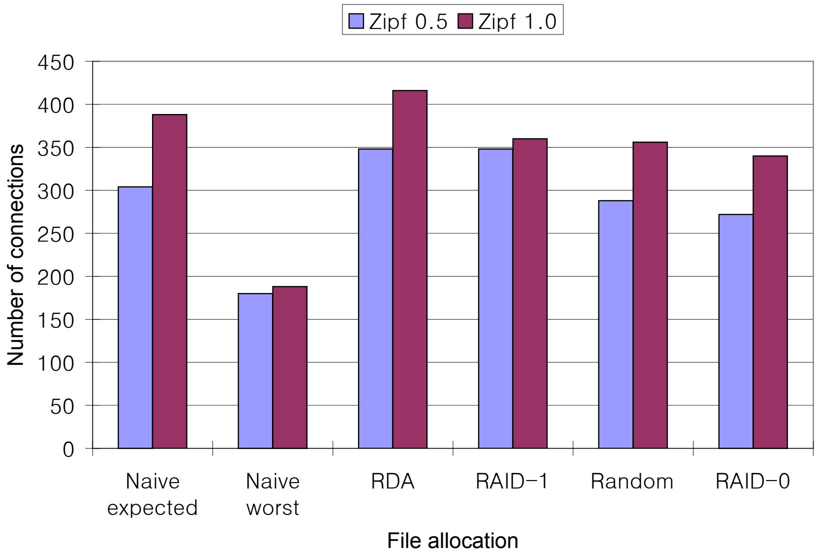

Figure 4. Performance under Zipf workload with parameter 0.5 and 1.0, measured in terms of number of simultaneous DVD-quality streams supported by storage server under various allocation policies.

6.3 Fail-Stutter Faults

Figure 6 represents tests that model single-disk fail-stutter faults that lead to reduced sequential transfer throughput. As discussed in Section 5, these tests model a disk with degraded bandwidth by having the server read additional useless data on every transfer from disk. The X axis represents the percentage of additional data transferred read from disk along with the request. The Y axis represents the number of times that client connections starved as a result of this bandwidth degradation (called client stoppages). The bars for each bandwidth degradation represent RAID-1, RDA, Naive, RAID-0, and random, from left-to-right. Confirming the observations of Arpaci-Dusseau, RAID-0 sees a large number of faults when one disk is slowed down, as it offers no protection against a single slow disk [2]. Random file allocation behaves even more poorly as bandwidth degrades since all blocks of all files are potential candidates for being allocated on the wrong disk. Interestingly, naive allocation does not perform as poorly as RAID-0 or random. This is because only those files that are actually allocated on the faulting disk will see timeouts; the other connections be-

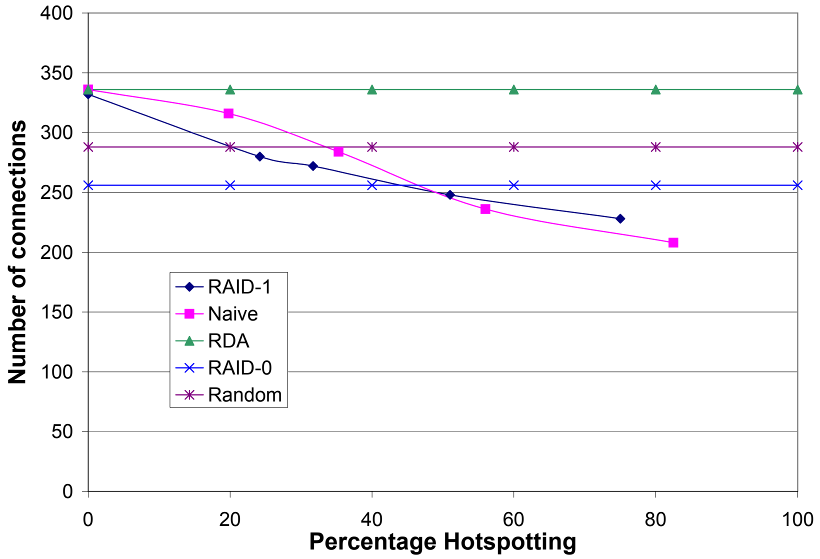

Figure 5. Performance of stream allocation policy when streams allocated on one disk are subject to an additional load beyond the uniform workload distribution.

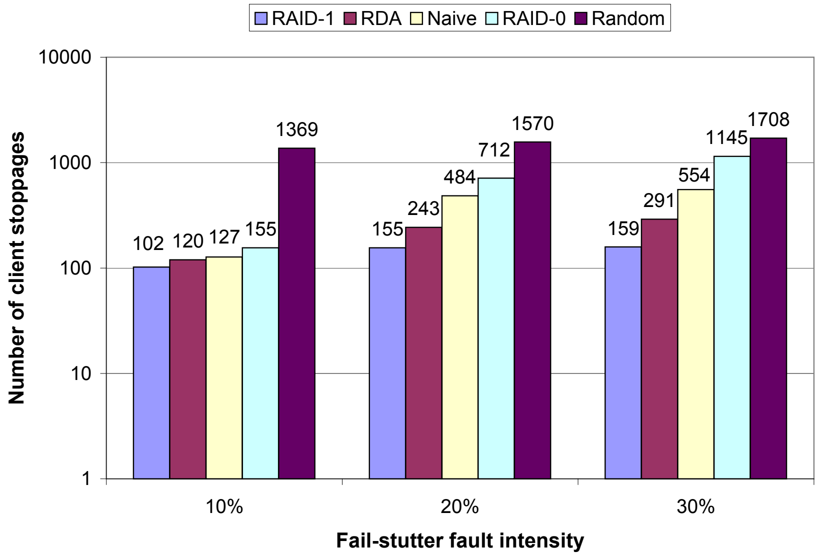

Figure 6. Number of times that client connections starve when one disk in the system is subject to fail-stutter faults that lead to degraded sequential transfer throughput. The X-axis represents the intensity of the fail-stutter faults modeled, while the Y-axis represents the number of times that client connections starve.

have completely normally. In contrast, RDA always has the choice to avoid the faulting disk, which it generally learns since the faulting disk will tend to have longer work queues than the other replica. This explains RDA's performance superiority to random, RAID-0, and naive. RAID-1 has slightly fewer connection stoppages than RDA because of a fault-isolation effect akin to that seen with naive (only files allocated to the bad mirror will be effected by the fault).

Figure 7 reports the average duration in seconds of a client connection starvation incident. RDA is the fastest to recover from client starvations, on the average, with RAID-1 and RAID-0 about equal, and naive and random substantially worse. RDA recovers promptly from client starvations since even if a particular block is chosen on the faulting disk, the very next block will probably not even have the faulting disk as a choice and thus is certain to be deliv-

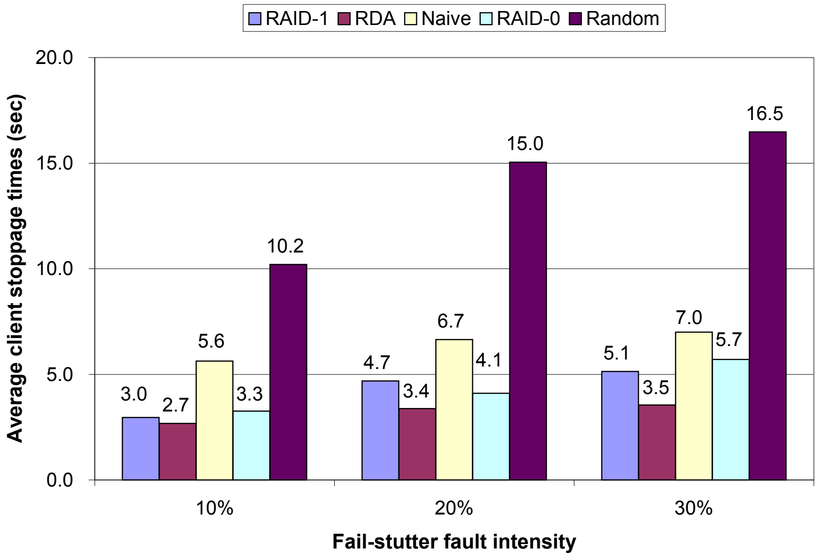

Figure 7. Average time in seconds of a client connection stoppage when one disk in the system is subject to fail-stutter faults that lead to degraded sequential transfer throughput. The X-axis represents the intensity of the fail-stutter faults modeled, while the Y-axis represents the average time of a client connection stoppage .

ered without delay. RAID-1, on the other hand, repeatedly sees the same choice of disks for every successive block in a stream, increasing the likelihood of one delay following another. RAID-0, naive, and random suffer because of a lack of choice in where to access any given block, causing the faulting disk to experience the same number of requests per unit time as any of the disks in normal operation. Since the faulting disk cannot keep up with this request load, it is likely to cause large delays. The same basic trends are seen for maximum client stoppage times in Figure 8, with the exception that naive is noticeably better than RAID-0 in the worst case.

6.4 Summary

Random duplicate allocation (RDA) is the only strategy studied that performs well in each of the workload hotspotting models when applied to disk arrays for DVD-quality video-on-demand storage servers. It also suffers less average and maximum client starvation time than any of the other schemes when subjected to single-disk fail-stutter faults that lead to bandwidth degradation, with a smaller number of total starvation incidents than all but one other allocation scheme (RAID-1). These results from an actual prototype storage server complement, experimentally confirm, and extend the prior theoretical analysis showing the advantages of RDA under simulated workload models [28].

7 Related Work

There have been numerous commercial implementations of VoD servers, as well as academic research on the subject. For example, Kasenna's Media Server uses a high-performance SCSI disk array to support 400 streams of rate 3 Mbps, while Bitband's Vision 680 supports 257 streams at 3.5 Mbps [5, 14]. Yang et al. have built a prototype server using custom hardware, achieving 550 simultaneous streams of rate 1.3 Mbps [36]. Shenoy and Vin studied storage techniques such as static and dynamic load balancing, retrieval techniques such as caching and batching, and disk striping policies based on detailed models of large disk arrays (16+ disks) with variable numbers of clients [30, 31]. Other works have considered VoD storage performance with disk striping and interleaving [21, 35]. There also have been several studies analyzing the characteristics of streaming media workloads [6, 32]. Prior studies have generally focused on bitrates below 5 Mbps or systems that support only a small number of distinct streams. More recent developments in link-level bandwidths, CPU performance, and disk technology trends require a careful investigation of faster bitrates, more connections, and more distinct streams.

Other works targeting VoD workloads have studied striping, replication, and randomization. Chua et al. propose a phase-based striping method and develop a data replication scheme to further reduce the average waiting time [8]. Using simulations, they showed that their striping method reduces waiting time and that replication further improves the performance by 25%. Chou et al. focus on the scalability of continuous media servers as a function of tradeoffs be-

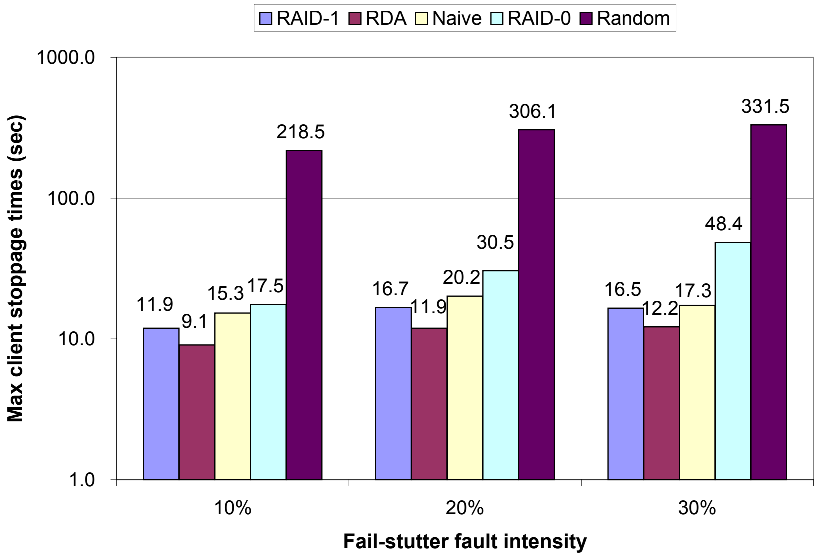

Figure 8. Maximum time in seconds of a client connection stoppage when one disk in the system is subject to fail-stutter faults that lead to degraded sequential transfer throughput. The X-axis represents the intensity of the fail-stutter faults modeled, while the Y-axis represents the maximum time of a client connection stoppage .

tween striping and replication [7]. Fu et al. compared the traditional block level randomization (BLR) techniques to what they call packet-level randomization (PLR) techniques using analytical models of server behavior [10]. They showed that PLR, which randomly allocates data on disks at the granularity of network packets (500–1400 bytes), can achieve superior short-term load balancing. However, PLR requires additional memory buffering since data must be simultaneously retrieved from all disks at the granularity of an allocation block (as much as 1 MB per disk). Santos et al. analytically compared random data allocation with traditional data striping techniques and found that random data allocation is at least as good as striping in streaming workloads [29]. Although all of the above works contribute to various aspects of video-on-demand storage design, our work extends upon the prior state-of-the-art by experimentally prototyping various file allocation strategies (including random duplicate allocation) and observing how those strategies respond to both workload hotspots and single-disk fail-stutter faults.

In networks, Rejaie et al. have provided a transport protocol for real-time multimedia streams [26], while Almeida et al. used multicast to service media workloads in education [1]. Issues related to network data delivery are important in the VoD context, but are orthogonal to the work presented here.

8 Conclusions

A disk array can provide high aggregate throughput, but only if the server can effectively balance the load on the disks. This paper investigates the performance of disk arrays for video-on-demand (VoD) workloads when subjected to workload hotspots and single-disk fail-stutter faults. This paper focuses on the random duplicate assignment (RDA) data allocation policy which places each data block on two disks chosen at random, independent of other blocks in the same media stream or other streams. This strategy is compared to traditional single-disk file allocation, disk striping (RAID-0), disk mirroring (RAID-1), and randomization without duplication. The various allocation schemes are implemented and tested using a prototype VoD server with 2 dual-core Opteron processors, 8 SATA disks, and 4 Gigabit Ethernet interfaces running the Linux 2.6 kernel.

The results indicate that combining randomization and replication allows RDA to effectively tolerate workload hotspots better than previous schemes. When exposed to a single-disk fail-stutter fault that results in bandwidth degradation, RDA sees fewer client starvation incidents than all allocation policies except for RAID-1. Although RDA sees somewhat more client starvation incidents than RAID-1, these starvation incidents tend to be shorter in duration because randomization implies that subsequent blocks from the same stream are unlikely to even be on the same faulting disk. These results from an actual prototype storage server complement and experimentally confirm the prior theoretical analysis showing the advantages of random duplicate allocation.

References

[1] J. M. Almeida, J. Krueger, D. L. Eager, and M. K. Vernon. Analysis of educational media server workloads. In *Proceed-*

ings of the 11th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV 2001), pages 21–30, June 2001.

[2] R. Arpaci-Dusseau. Performance availability for networks of workstations. *PhD thesis, University of California, Berkeley*, 1999.

[3] R. H. Arpaci-Dusseau and A. C. Arpaci-Dusseau. Fail-stutter fault tolerance. *Proceedings of the Workshop on Hot Topics in Operating Systems - HOTOS*, pages 33 – 38, 2001.

[4] R. D. Barve, E. F. Grove, and J. S. Vitter. Simple randomized mergesort on parallel disks. *Parallel Computing*, 23(4):601–631, 1997.

[5] Bitband Technologies Ltd. Vision 680. Data Sheet.

[6] M. Chesire, A. Wolman, G. Voelker, and H. Levy. Measurement and analysis of a streaming-media workload. *Proceedings of 3rd USENIX Symposium on Internet Technologies and Systems. (USITS'01)*, pages 1 – 12, 2001.

[7] C.-F. Chou, L. Golubchik, and J. Lui. Striping doesn't scale: how to achieve scalability for continuous media servers with replication. *Proceedings 20th IEEE International Conference on Distributed Computing Systems*, pages 64 – 71, 2000.

[8] T. S. Chua, J. Li, B. C. Ooi, and K.-L. Tan. Disk striping strategies for large video-on-demand servers. *Proceedings ACM Multimedia 96*, pages 297 – 306, 1996.

[9] Entone Technologies, Inc. Entone Video Server Architecture. White paper, 2005.

[10] K. Fu and R. Zimmermarm. Randomized data allocation in scalable streaming architectures. *Database Systems for Advanced Applications. 10th International Conference, DASFAA 2005. Proceedings*, pages 474 – 86, 2005.

[11] G. R. Ganger, B. L. Worthington, R. Y. Hou, and Y. N. Patt. Disk subsystem load balancing: Disk striping vs. conventional data placement. In *Proceedings of the Twenty-Sixth Annual Hawaii International Conference on System Sciences*, volume I, pages 40–49, January 1993.

[12] G. H. Gonnet. Expected length of the longest probe sequence in hash code searching. *Journal of the Association for Computing Machinery*, 28(2):289–304, April 1981.

[13] E. Grochowski and R. D. Halem. Technological impact of magnetic hard disk drives on storage systems. *IBM Systems Journal*, 42(2):338–346, April 2003.

[14] Kasenna, Inc. Kasenna Media Servers. Data Sheet, August 2003.

[15] J. R. Larus and M. Parkes. Using Cohort Scheduling to Enhance Server Performance. In *Proceedings of the 2002 USENIX Annual Technical Conference*, pages 103–114, June 2002.

[16] S.-H. Lim, Y.-W. Jeong, and K.-H. Park. Interactive media server with media synchronized raid storage system. In *NOSSDAV '05: Proceedings of the international workshop on Network and operating systems support for digital audio and video*, pages 177–182, New York, NY, USA, 2005. ACM Press.

[17] E. P. Markatos, M. Katevenis, D. N. Pnevmatikatos, and M. Flouris. Secondary Storage Management for Web Proxies. In *USENIX Symposium on Internet Technologies and Systems*, pages 93–104, October 1999.

[18] L. W. McVoy and S. R. Kleiman. Extent-like Performance from a UNIX File System. In *Proceedings of the USENIX Winter 1991 Technical Conference*, pages 33–43, Dallas, TX, USA, 1991.

[19] R. V. Meter. Observing the effects of multi-zone disks. In *Proceedings of the USENIX 1997 Annual Technical Conference*, pages 19–30, January 1997.

[20] E. M. Nahum, T. Barzilai, and D. Kandlur. Performance Issues in WWW Servers. *IEEE/ACM Transactions on Networking*, 10(2):2–11, February 2002.

[21] B. Özden, R. Rastogi, and A. Silberschatz. Disk striping in video server environments. In *Proceedings of the Third IEEE International Conference on Multimedia Computing and Systems*, pages 580–589, June 1996.

[22] V. S. Pai, P. Druschel, and W. Zwaenepoel. Flash: An Efficient and Portable Web Server. In *Proceedings of the USENIX 1999 Annual Technical Conference*, pages 199–212, June 1999.

[23] V. S. Pai, P. Druschel, and W. Zwaenepoel. I/O-Lite: A Unified I/O Buffering and Caching System. In *Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation*, pages 15–28, February 1999.

[24] D. A. Patterson, G. Gibson, and R. H. Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID). In *Proceedings ACM SIGMOD Conference*, Chicago, Illinois, June 1988.

[25] A. N. Reddy and P. Banerjee. An Evaluation of Multiple-Disk I/O Systems. *IEEE Transactions on Computers*, 38(12):1680–1690, December 1989.

[26] R. Rejaie, M. Handley, and D. Estrin. RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the internet. In *Proceedings of IEEE INFOCOM '99*, pages 1337–1345, March 1999.

[27] A. Rousskov and D. Wessels. The Third Cache-off. Raw data and independent analysis at http://www.measurement-factory.com/results/, October 2000.

[28] P. Sanders, S. Egner, and J. Korst. Fast concurrent access to parallel disks. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, volume 11, pages 849–858, San Francisco, January 2000.

[29] J. Santos, R. Muntz, and B. Ribeiro-Neto. Comparing random data allocation and data striping in multimedia servers. *Performance Evaluation Review*, 28(1):44 – 55, 2000.

[30] P. Shenoy and H. Vin. Multimedia storage servers. In K. Jeffay and H. Zhang, editors, *In Readings in Multimedia Computing and Networking*. Morgan Kaufmann Publishers, 2002.

[31] P. Shenoy and H. M. Vin. Efficient Striping Techniques for Variable Bit Rate Continuous Media File Servers. *Performance Evaluation Journal*, 38(3):175–199, December 1999.

[32] K. Sripanidkulchai, B. Maggs, and H. Zhang. An analysis of live streaming workloads on the internet. *Proceedings of the 2004 ACM SIGCOMM Internet Measurement Conference, IMC 2004*, pages 41 – 54, 2004.

[33] J. S. Vitter. External memory algorithms and data structures: Dealing with massive data. *ACM Computing surveys*, 33(2):209–271, June 2001.

[34] M. Welsh, D. Culler, and E. Brewer. SEDA: An architecture for well-conditioned, scalable internet services. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles*, pages 230–243, October 2001.

[35] W. E. Wright. An efficient video-on-demand model. *IEEE Computer*, pages 64–70, May 2001.

[36] S. Yang, H. Yang, and Y. Yang. Architecture of high capacity vod server and the implementation of its prototype. *IEEE Transactions on Consumer Electronics*, pages 1169–1177, November 2003.