A Model and Prototype of a Resource-Efficient Storage Server for High-Bitrate Video-on-Demand [*]

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

Abstract

This paper presents a mathematical model and a prototype of a resource-efficient storage server for high-bitrate video-on-demand (VoD) applications. Rapid exponential growth of disk capacity enables the storage of high-bitrate VoD streams; however, a server system must be carefully designed to allow those streams to be retrieved from disk and delivered to the network efficiently. Additionally, a cost-effective server should be implemented using only commodity components, such as standard PCs, SATA disks and controllers, and Gigabit Ethernet links.

Previous parallel I/O performance models have been either oversimplified theoretical models that ignore hardware and application characteristics or complex hardware models that consider detailed disk behaviors such as inter-track variations. This paper presents a model between these extremes: detailed enough to account for the rate-based nature of streaming video, the buffering time allowed by the application, and average-case disk hardware characteristics while remaining simple enough to use for algorithm and system design. This paper then describes a prototype storage server designed to serve large video files at the specified bitrates and finds its performance to agree closely with the model (with an average discrepancy of 11% for high-bitrate streams). The system uses up to 8 SATA-300 disks and can simultaneously serve 290 distinct DVD-quality (6 Mbps) streams or 74 distinct HDTV-quality (25 Mbps) streams from disk, achieving an aggregate network throughput of 1.85 Gbps.

1 Introduction

As the computational capability and network bandwidth available to end-users has increased dramatically, all levels have seen demand for richer and more targeted content. For example, the mobile space has seen the arrival of the video iPod, as well as services such as Verizon V-CAST that bring video to cell phones. High-end entertainment has seen a rapid increase in pay-per-view and 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. All of these trends indicate an increased need for efficiently serving on-demand video.

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 [12]. 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.

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, 19, 22, 23, 35]. 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 [16]. 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].

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

Theoretical computer science has also produced advances in parallel disk models and algorithms, but these do not account for real hardware or software characteristics [34].

The goal of this work is to model and prototype a resource-efficient storage server for VoD applications with bitrates ranging from 1 Mbps (roughly the quality viewable on an iPod) to 25 Mbps (HDTV-quality), with a key operating point at 6 Mbps (DVD-quality). The server should be built using commodity hardware and as few disks as possible while also serving as many high-bitrate streams as allowed by the network links. The design should be based on sound principles resulting from the model-based analysis.

The contributions of this paper are twofold. First, this paper presents a performance model of a VoD storage server that accounts for the rate-based nature of streaming video, the buffering time allowed by the application, and the disk performance characteristics. The model is a substantial improvement over existing parallel I/O models both for its ease of use and for its ability to capture realistic hardware and application characteristics. Second, this paper describes a prototype storage server designed to serve large video files at specified bitrates and finds its performance to agree closely with the model, with an average discrepancy of only 11% for high-bitrate streams. The system uses up to 8 SATA-300 disks and can simultaneously serve 290 distinct DVD-quality (6 Mbps) streams or 74 distinct HDTV-quality (25 Mbps) streams from disk, achieving an aggregate network throughput of 1.85 Gbps.

2 Modeling a VoD Storage Server

System design is a challenging process and can be aided by the use of models. Models are ideally detailed enough to capture important performance effects while still simple enough to use easily. The following gives a model for designing a parallel I/O system suited for delivering VoD from a modern disk array.

Parallel I/O. A popular model for theoretical analysis of parallel disk I/O systems is the parallel disk model (PDM) [34]. PDM accounts for a fixed-size memory buffer and assigns a unit cost for each disk access, casting the optimization problem as one of minimizing the number of atomic I/O steps. This model has been extensively used for analysis and optimization of external-memory applications under adversarial request models [1, 5, 13, 30]. Further, Barve et al. have provided a competitive online algorithm for retrieving a single multimedia reference stream from multiple disks [4]. However, the general problem of optimally scheduling multiple streams on parallel disks has been proven NP-Complete [2]. As a further complication, the model does not account for the variable latencies of real disks that consist of seek time, rotational latency, and transfer time, or for realistic application characteristics such as real-time delivery.

Disk characteristics. Seek time and rotational latency are independent of transfer size, but the transfer time depends on the transfer size used to access the disk and the disk's sequential transfer rate. Consequently, disks can be accessed more efficiently if the block sizes are large enough to amortize the overhead of seeking and rotational latencies. However, using exclusively large block sizes could be extremely wasteful when storing small pieces of data (e.g., typical Unix files with a median size of 10 KB [18]).

Rotational latencies between discontiguous disk accesses are uniformly distributed with an average of $\frac{30000}{R}$ milliseconds, where R is the rotational speed of the disk stated in RPMs. However, neither seek time nor sequential transfer rate is a constant; seeks that must travel a short distance are much faster than those that span the disk, and transfers at outer tracks provide higher bandwidth because of the higher linear velocity of the magnetic medium. Properly considering the distributions of these terms in detail requires extensive knowledge of the layout and workload that may not be available at design time. Ruemmler and Wilkes have designed a disk simulation model and explain how various disk drive performance components can affect its accuracy [28]. They quantify the effectiveness of their model by comparing performance distribution curves for a real drive and the model output; they use the root mean square of the horizontal distance between these two curves as a metric called the *demerit* figure. Achieving a truly detailed model with a very low demerit figure requires a detailed understanding of the seek time, rotational positioning, disk data layout, and data caching characteristics of the disk. Although this level of information may be possible under certain circumstances, it is often difficult or impossible to extract from commodity operating systems and disk drives. Such detailed information is also difficult to acquire before building the actual system, making such a model less valuable for performance-oriented design. Consequently, the analysis presented here chooses to abstract out those details and only considers the average seek times and sequential transfer rates. (Note that even these measures incorporate some short seeks, as these are present in sequential transfers of large blocks.)

If the average access time (seek plus rotational latency) is termed t_a and the average sequential transfer rate is termed r_d, the average time to access a contiguous block of disk with size B is $t_a + \frac{B}{r_d}$. Typical values of t_a and r_d range from 17 milliseconds and 440 Mbps for high-capacity SATA drives to 6 milliseconds and 800 Mbps for high-performance SCSI drives. These values may be measured using tools such as lmbench [17].

Multimedia performance model. Real multimedia servers such as video-on-demand must schedule N simultaneous connections onto their D disks. Each connection is transferring content from a large logically contiguous file with a target bitrate (r_c). Additionally, the server may only force each connection to buffer data for a certain amount of time, as the buffer time affects the initial startup delay of viewing the stream or the amount of time required for recovery after a fast-forward or rewind operation. We call this time t_b; based on user perceptions of Web quality of service, this time should be no more than 5–10 seconds [7].

We may define a measure of *disk availability* as the number of disks available over a period of time. Because the server has D disks, there are simply Dt_b units of disk availability during t_b amount of time. At the same time, each of the N simultaneous streams must retrieve $r_c t_b$ data in that time period to keep its client connection active; $r_c t_b$ is the conceptual block size of logical data retrieval in this system. Using the above formula for disk access time, each stream requires $t_a + \frac{r_c t_b}{r_d}$ units of disk availability in t_b time. We then solve for N by comparing the required disk availability and the total disk availability.

$$N(t_a + \frac{r_c t_b}{r_d}) = Dt_b \;\Rightarrow\; N = \frac{Dt_b}{t_a + \frac{r_c t_b}{r_d}} = \frac{Dt_b r_d}{t_a r_d + r_c t_b}$$

This represents the maximum number of simultaneous distinct streams available from the disk subsystem, assuming no contention for resources when accessing the disks. Alternatively, solving for D gives $D = N\frac{t_a r_d + t_b r_c}{t_b r_d}$ as the number of disks required to support such a number of streams. Since the streams must then be delivered on the network, N is also bounded by $\frac{r_l}{r_c}$ where r_l is the achievable link-level bitrate (949 Mbps for TCP/IP over Gigabit Ethernet.)

Accounting for RAID. A common approach to improving disk bandwidth for certain classes of applications is to use RAID to stripe data across disks [24]. In particular, each large sequential transfer will be striped across several disks depending on the RAID level used. The disk access (with cost t_a) will take place simultaneously across disks, but the transfer rate will be multiplied by the number of disks across which it has been striped, termed D_s. Note that D_s only considers the number of disks across which a given transfer has been striped, not the total number of disks in the array. In total, a sequential transfer of size $r_c t_b$ takes $t_a + \frac{r_c t_b}{r_d D_s}$ time, using $t_a D_s + \frac{r_c t_b}{r_d}$ units of disk availability. The disk usage corresponding to the access time has been multiplied but that from the transfer time is unchanged. Solving for N now gives:

$$N = \frac{Dt_b}{t_a D_s + \frac{r_c t_b}{r_d}} = \frac{Dt_b r_d}{t_a D_s r_d + r_c t_b}$$

Since this is strictly less than the non-striped version (and degrades as D_s increases), this model predicts better peak performance without RAID than with it. This is somewhat contrary to conventional wisdom but jibes with the observation of Reddy and Banerjee that disk striping increases the effective service time of a request [25]. A possible solution to avoid this increase is to make the disk stripe size at least as large as typical requests, thus giving a D_s of 1 for such transfers [11]. However, a server that supports multiple bitrates invariably has multiple reasonable request sizes; choosing the smallest leads to a large D_s for larger transfers, while choosing the largest gives no bandwidth improvement for small transfers. Consequently, the prototype server studied here manages the disks in the array as independent units rather than as a RAID.

3 Prototype Implementation

We have built and started to experiment with a prototype streaming storage server. The following discusses the hardware and software used in this study.

Hardware platform. This study uses two hardware systems, one based on a SATA-1 disk array and another based on a SATA-300 disk array. The SATA-1 platform uses a Promise FastTrak TX4200 4-port SATA RAID controller, a motherboard with a 2-port Intel SATA controller, 1–6 Seagate SATA-1 disks with capacity 400 GB and 7200 RPM rotation speed, two 2.8 GHz Pentium-4 Xeon hyperthreaded processors, two Gigabit Ethernet links, and 4 GB of DRAM. The SATA-300 platform uses a Promise FastTrak SX8300 8-port SATA-300 RAID controller, 1–8 Seagate SATA-300 disks with capacity 500 GB and 7200 RPM rotation speed, two 2.0 GHz dual-core Opteron processors, two Gigabit Ethernet links, and 4 GB of DRAM. As discussed in Section 2, this study configures the disks independently rather than as a RAID.

The SATA-1 disks achieve a disk access time (t_a) of 17 ms and a sequential transfer rate (r_d) of 525 Mbps. Although SATA uses a point-to-point link for each disk, the controller is on a PCI bus that is limited to 2.1 Gbps. Consequently, inter-disk contention becomes an issue with 4 or more disks. With 4 disks simultaneously running, the per-disk r_d is 431 Mbps; with 6 disks (placing the last two on the motherboard controller), the per-disk r_d is 458 Mbps. The SATA-300 disks see t_a of 16 ms, r_d of 446 Mbps (because of their larger size), and negligible inter-disk contention because the controller has an 8 Gbps PCI-X host interface.

Server software. The software platform consists of a standard Linux 2.6.8 kernel using the ext3 filesystem for all data partitions. Because the operating system is unchanged, it may still allocate disk blocks at a minimum granularity of 4K; in practice, the ext3 filesystem tends to allocate file blocks sequentially. The content is stored on disk as large files, with one stream data partition per disk. The SATA-1 disk array is large enough to hold over 500 4.7 GByte DVD-quality streams, while the SATA-300 array can hold over 800. However, even these numbers are not enough to fully

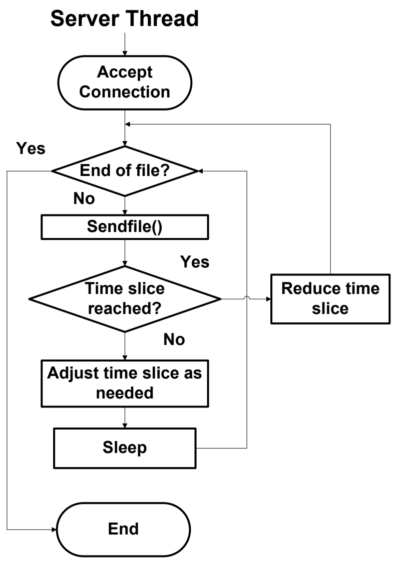

Figure 1. Flowchart describing operations in server thread for processing individual video-on-demand (VoD) client connection

exercise the system at low rates. Consequently, these tests use an even larger number of 1 GByte streams. No stream data is reused in the filesystem cache across performance tests because each test overflows the cache in less than one minute.

Figure 1 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 $r_c t_b$ data as an initial buffer. It then enters a repeated pattern of sending $r_c t_b$ data, seeing how long it took to perform that transmission, and sleeping that connection for the remainder of t_b. If a transmission takes longer than t_b, 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 t_b, the server adds the excess time to a per-connection deficit variable (t_d) and sets the target time for the next transmission of $r_c t_b$ data to be $t_b - t_d$. 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 performance tests use a t_b of 5 seconds and report a prototype N value as the maximum number of simultaneous connections to distinct streams that can be successfully supported by the server. 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. The client connections come from one or two client machines on the same LAN as the server, each initiating parallel requests for distinct content. 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.

4 Experimental Results

Figures 2 and 3 show the number of simultaneous distinct streams that can be served by the SATA-1 and SATA-300 systems for high bitrates. From bottom to top, the graphs represent the performance of systems with 1, 2, 4, 6, and (for SATA-300) 8 disks. In each case, one set of bars shows the prototype at each of the target bitrates: 6, 12, 18, and 25 Mbps, while the other set shows the performance predicted by the model (saturating at 1.9 Gbps because of link-level bandwidth limits). The model graphs are next to the prototype result graphs for comparison. The model graphs assumes the same r_d as in the 1 disk case except for SATA-1 with 4 and 6 disks. In these two cases, the model graphs account for inter-disk contention by using the adjusted r_d values given in Section 3. None of the other configurations see noticeable inter-disk contention.

For the bitrates shown in Figures 2 and 3, the model closely predicts the performance actually achieved by the prototype. The average discrepancy is only 11% across all combinations of disk array size and bitrate. These results serve to validate the model and show that it can be used for effective system design.

Further analysis shows that in both systems, limitations from the operating system and PCI bus cap the actual network throughput to 1.85 Gbps; this measurement was taken by sending data from memory rather than accessing disk. This limit explains why the SATA-300 prototype does not scale to the expected 1.9 Gbps link-level bandwidth; the 8-disk SATA-300 system actually saturates the network for 18 and 25 Mbps streams. The remaining discrepancies may stem from other limitations not considered by the model, such as memory transfer bandwidth, I/O bus contention from other devices, CPU utilization, or seek time and transfer rate variations across the disk.

Figure 4 shows the performance achieved by the prototype and predicted by the model for 1 Mbps streams. The bars are similar to those used in Figures 2 and 3, representing the number of simultaneous connections to distinct streams as the number of disks is varied. The prototype values are within 15% of the model when only 1 or 2 disks are used in the SATA-1 system or with 1–4 disks in the SATA-300 system. However, the numbers start to diverge substantially as more disks are used. This discrepancy arises

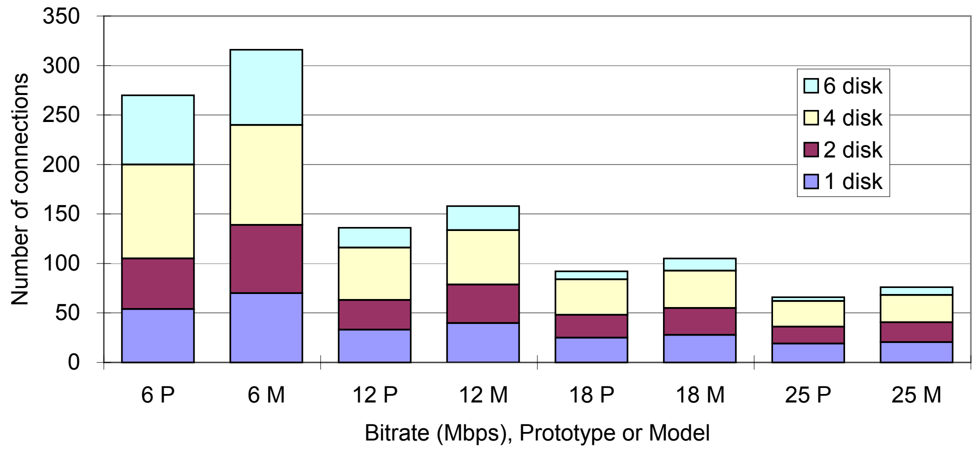

Figure 2. Performance of high bitrate SATA-1 system according to model and prototype.

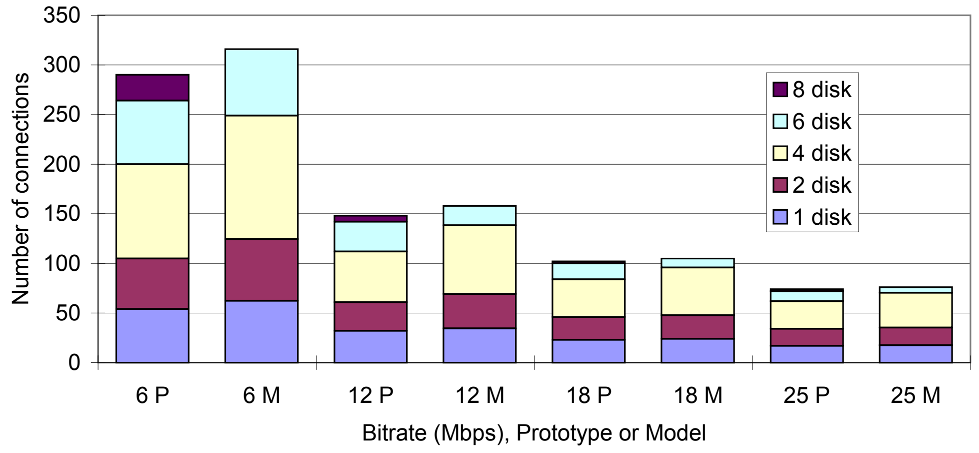

Figure 3. Performance of high bitrate SATA-300 system according to model and prototype.

because the prototype uses a software concurrency architecture with one thread per connection, and CPU utilization becomes saturated with about 400 threads per processor. We are currently investigating alternative software architectures to achieve sufficient concurrency without a large number of threads.

5 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 [8, 14]. Yang et al. have built a prototype server using custom hardware, achieving 550 simultaneous streams of rate 1.3 Mbps [37]. 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 [31, 32]. Chang and Garcia-Molina focused on the best memory use techniques under different disk management policies [10]. There have also been numerous studies in disk-scheduling algorithms such as EDF, CScan, and Real-Time [9, 20, 36]. Other works have considered VoD storage performance with disk striping and interleaving [6, 21, 36]. Wright describes and analytically evaluates a scheme called one-way elevator with interleaving and delayed start (OEID); this scheme interleaves data over all disks and delays the start of each new stream to reduce random fluctuations in disk load caused by new arrivals [36]. In general, 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.

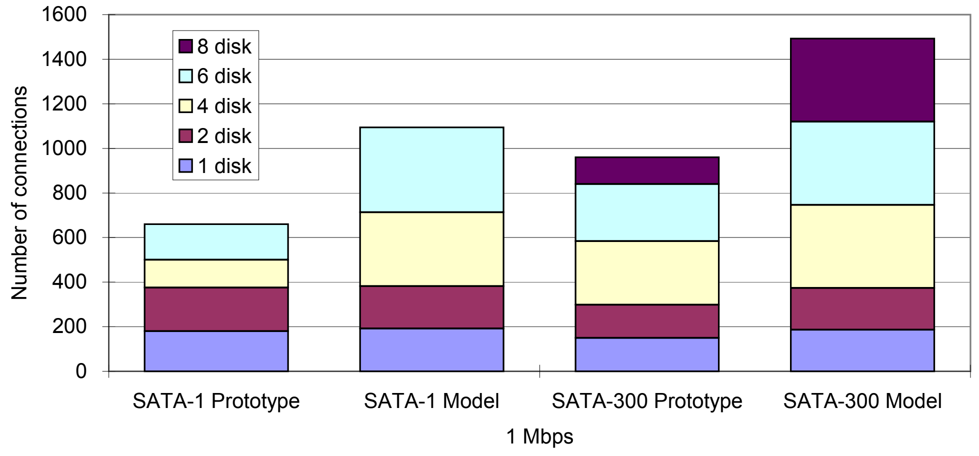

Figure 4. Performance of 1 Mbps SATA-1 and SATA-300 system according to model and prototype.

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 [3]. Such protocol issues are orthogonal to the design studied in this paper and may be combined productively with this work.

6 Conclusions and Discussion

This paper contributes to the state-of-the-art in parallel I/O performance modeling by providing a model that is simple enough to use for algorithm and system design while also being detailed enough to account for streaming application characteristics and disk hardware trends. This paper then describes a prototype storage server designed to serve large video files at bitrates ranging from 1–25 Mbps and finds its performance to agree closely with the model (with an average discrepancy of only 11% for high-bitrate streams). The prototype can deliver 1.85 Gbps of aggregate network throughput when serving distinct streams from disk, using a commodity PC-based server, an unmodified operating system, and inexpensive SATA disks. These results are promising by indicating that commodity equipment can be used to support emerging high-bitrate streaming media and that such systems can be designed and evaluated analytically.

One workload-related limitation of the model of Section 2 is that it targets only the maximum possible performance for delivering data from a VoD storage server's disk array. This performance level is achieved when N distinct streams are uniformly spread across D disks. The model should be expanded to consider other workloads. For example, some reuse patterns would allow successful file caching, reducing the disk utilization. On the other hand, hotspotting could reduce the effective parallelism of a disk array. Analysis has shown that possible solutions to avoid hotspotting such as RAID or random placement of disk blocks may still lead to a loss of concurrency and performance under certain workloads [5, 33]. Sanders et al. gave an elegant technique to avoid disk hotspotting by placing each logical disk block (in our case, a chunk of at least $r_c t_b$) on two randomly selected disks so that at least one copy can be read without I/O bottlenecks [29]. Such a strategy trades off some capacity for more reliable performance, but the current rate of capacity growth may make this strategy a reasonable tradeoff.

References

[1] A. Aggarwal and J. S. Vitter. The Input/Output complexity of sorting and related problems. *Communications of the ACM*, 31(9):1116–1127, 1988.

[2] A.Gulati and P. Varman. Scheduling Multiple Flows on Parallel Disk Systems. In *Proceedings of the International Conference on High-Performance Computing (HiPC '05)*, December 2005.

[3] J. M. Almeida, J. Krueger, D. L. Eager, and M. K. Vernon. Analysis of educational media server workloads. In *Proceedings of the 11th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV 2001)*, pages 21–30, June 2001.

[4] R. Barve, M. Kallahalla, P. Varman, and J. S. Vitter. Competitive Parallel Disk Prefetching. *Journal of Algorithms*, pages 152–181, 2000.

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

[6] S. Berson, S. Ghandeharizadeh, R. Muntz, and X. Ju. Staggered striping in multimedia information systems. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 23(2):79 – 90, 1994.

[7] N. Bhatti, A. Bouch, and A. Kuchinsky. Integrating User-Perceived Quality into Web Server Design. In *Proceedings of the Ninth International World Wide Web Conference*, 2000.

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

[9] P. Bosch and S. Mullender. Real-time disk scheduling in a mixed-media file system. *Proceedings Sixth IEEE Real-Time Technology and Applications Symposium. RTAS 2000*, pages 23 – 32, 2000.

[10] E. Chang and H. Garcia-Molina. Effective memory use in a media server. *Proceedings of the Twenty-Third International Conference on Very Large Databases*, pages 496 – 505, 1997.

[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] 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.

[13] M. Kallahalla and P. J. Varman. Optimal read-once parallel disk scheduling. In *In Proc. of Sixth ACM Workshop on I/O in Parallel and Distributed Systems*, pages 68–77, 1999.

[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] 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.

[17] L. McVoy and C. Staelin. lmbench: Portable Tools for Performance Analysis. In *Proceedings of the 1996 USENIX Technical Conference*, pages 279–295, January 1996.

[18] S. J. Mullender and A. S. Tanenbaum. Immediate files. *Software: Practice and Experience*, 14(4):365–368, April 1984.

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

[20] A. Narasimha Reddy and J. Wyllie. I/o issues in a multimedia system. *Computer*, 27(3):69 – 74, March 1994.

[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] C. Ruemmler and J. Wilkes. An introduction to disk drive modeling. *IEEE Computer*, 27(3):17 – 28, March 1994.

[29] 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.

[30] R. Shah, P. J. Varman, and J. S. Vitter. Online algorithms for prefetching and caching on parallel disks. In *Proceedings of the ACM Symposium on Parallelism in Algorithms and Architectures*, 2004.

[31] 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.

[32] 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.

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

[34] J. S. Vitter and E. A. M. Shriver. Optimal algorithms for parallel memory I: Two-level memories. *Algorithmica*, 12(2-3):110–147, 1994.

[35] 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.

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

[37] 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.