# Algorithmic and Implementations of Storage Networks

Kay Salzwedel[1]

HEINZ NIXDORF INSTITUTE
Paderborn, Germany
nkz@upb.de

**Abstract.** The last storage hierachy in modern computers has to consists of durable, permanent storage devices that can keep the data over a long period of time. Obviously, this hierachy has the slowes access time and the larges capacity. Nowadays, magnetic hard disks are used to store the large amounts of data permanently. But even the ever growing capacity of disk drives[1] cannot satisfy the ever growing demands for flexible and efficient storage systems. The simplest way to provide almost indefinite storage capacity is the use of disk drives in parallel. The underlying architecture of such a *storage network* can vary significantly. All we require is a number of storage devices (e.g. disk drives) that are connected and can be accessed from the outside.
In this paper we give an overview of existing techniques and implementations tackling the problems arising in such systems. Starting point will be an introduction of the problems originated in storage systems, like data distribution or data access balance. This leads directly to the definition and discussion of quality factors imposed on these networks. Using these factors will enable us to compare rather different systems even if a practical comparison is not possible. We describe the main ideas underlying the introduced systems and analyse them using the defined quality measures.

## 1 Introduction

Persistent storage in modern computers is handled by magnetic hard disk drives. They form the last, and therefore the slowest level in the memory hierachy. Disk drive technology is very sophisticated and provides an accelerated growth in disk capacity. Nowadays, a single of-the-self drive is capable of storing approximately 80 GB and the capacity is doubled almost every 14 – 18 month. Nevertheless, this is not sufficient to manage the ever growing data volumes. The fast

[1] Today, of-the-shelf disks fast approach capacities of 100 GB and this number is doubled almost every 14 – 18 month.

growing processing power of modern computers, the global availability of information, and the increasing use of mixed-media contents demand for more flexible storage systems. The simplest solution of using many disk drives independently seems to solve the capacity problem but does not scale if the number of data requests and number of used disks are concerned, i.e. a very popular data element may be stored on one disk and many users want to access it, resulting in hot spots and very slow data delivery.

The above reasoning implies that a number of problems have to be addressed until a larger collection of disk drives satisfies the imposed needs of users. We define a *storage network* as a collection of disks connected by an arbitrary network. The underlying topology or architecture of the network is unrestricted. It may vary from disks attached to a SCSI bus up to very complex networks like the Internet where the disks drives are associated with server nodes. Having a closer look at this abstraction, two crucial observations become apparent. Larger demands for capacity are easily solved by attaching more disks to the network. Nevertheless, the data need to be distributed in an intelligent way to provide an even utilization of disks. Only this distribution can ensure the scalability in term of data requests when the number of disk drives is increased. Furthermore, it offers the possibility of a parallel data access provided that the data resides on different disk drives. This improves not only the acces time but also the utilization because large data requests are served by more then one disk.

But using a storage network imposes another inherent restriction – **availability**. Every disk has an estimated operation time during which an error is rather unlikely to occur. This entity is called *mean time to/between failure* (MTTF or MTBF, respectively) and for modern drives this number is in the order of 1,000,000 hours. Regarding a whole storage network, the MTBF decreases significantly. This is due to the fact, that it depends on the failure rate $\lambda_{sys}$ [18, 8] which is simply the sum of the failure rates $\lambda_i$ of each participating disk $D_i$[2]. This problem can be solved by using redundancy of some kind. The easiest way is to store $k$ copies of all data like in an RAID

[2] This is only true under the assumtions that the failure rate is constant which is equal to an exponentially distributed time to failure and the failures are independent of each other. For all systems considered in this paper these assumptions are fulfilled.

1 system (with $k = 1$) [11] or the PAST storage server [12, 7]. In case of a disk failure, the system is still operational and the faulty disk can be replaced. Because the copying of data results in a large waste of resources, more space efficient methods are sought. One of the common approaches is the use of parity information like in RAID 4/5 [11, 5], Random RAID [3], Swarm [10, 9], and many video servers [2, 13, 19, 20, 1]. The idea is to split a data object into $l$ subblocks and derive an additional subblock $l + 1$ which stores the result of an XOR operation over all subblocks. Assuming the subblocks reside on different disks, one disk failure can be tolerated and all data can be rebuild by accessing the remaining $l$ subblocks. This technique can be expanded to allow multiple disk faults by deriving multiple parity information [4, 15, 16, 17].

The second problem is the *data distribution problem*. It has to give an answer to the following question. Where should the data be stored so that any data request can be answered as fast as possible without forming hot spots? Furthermore, the data distribution has to ensure *scalability*. The more disks are in a storage network the more data requests should be possible or the faster should the data be delivered. This can only be done if the data is distributed so that many disks participate when delivering data. There are a large number of different approaches including deterministic striping, e.g. [11, 6, 2] and randomized placement, e.g. [1, 3, 14].

The above reasoning can be summerized be defining a number of properties a storage network has be concerned with.

1. **Availability:** Because of the increased sensitivity to disk faults storage networks need to implement some kind of fault tolerant mechanism to tolerate the loss of some disk drive.
2. **Space and Access Balance:** To ensure good disk utilization and provide scalability the data should be evenly distributed over all disk drives. The data requests are usually generated by a file system above the storage network. Hence, the storage network has no knowledge about the access pattern and must handle any possible request distribution.
3. **Resource Efficiency:** Redundancy implies a necessary waste of resource. Nevertheless, systems should use all its resources in an useful way.

4. **Access Efficiency:** Stored data that cannot be accessed is lost data. This property defines *time* and *space* requirements needed to access an arbitrary data element. These requirements are of crucial importance when very large networks are concerned.
5. **Heterogeneity:** Hetereogeneity describes the possibility to handle disks of different size and different speed.
6. **Adaptability:** The information volume is constantly growing and even generous planned system can quickly run out of space. This property measures the ability to adapt to a changed demand like capacity or performance.
7. **Locality:** Locality describes the degree of communication required to answer any data request. It distinguished between centralized and distributed approaches.

It seems impossible for a storage network to meet all the above requirements. But almost all of the described storage networks are developed to solve at least one of them. In the rest of this paper we will overview many very different systems by describing their solutions to the above requirements.

## References


[1] J. Alemany and J. S. Thathachar. Random striping news on demand servers. Technical Report TR-97-02-02, University of Washington, Department of Computer Science and Engineering, February 1997.

[2] S. Berson, R. Muntz, S. Ghandeharizadeh, and Xiangyu Ju. Staggered striping in multimedia information systems. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 23(2):79–90, June 1994.

[3] Yitzhak Birk. Random RAIDs with selective exploitation of redundancy for high performance video servers. In *Proceedings of 7th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'95)*, April 1997.

[4] Mario Blaum, Jim Brady, Jehoshua Bruck, and Jai Menon. EVENODD: an optimal scheme for tolerating double disk failures in RAID architectures. In *Proceedings of the 21st Annual International Symposium on Computer Architecture*, pages 245–254, Chicago, Illinois, April 18–21, 1994. IEEE Computer Society TCCA and ACM SIGARCH.

[5] Peter M. Chen, Edward K. Lee, Garth A. Gibson, Randy H. Katz, and David A. Patterson. RAID: high-performance, reliable secondary storage. *ACM Computing Surveys*, 26(2):145–185, June 1994.

[6] Peter Ming-Chien Chen and David A. Patterson. Maximizing performance in a striped disk array. Technical Report CSD-90-559, University of California, Berkeley, 1990.

[7] Peter Druschel and Antony Rowstron. PAST: A large-scale, persistent peer-to-peer storage utility. In *Hot Topics in Operating Systems (HotOS VIII)*, Schloss Elmau, Germany, May 2001.

[8] Garth Alan Gibson. *Redundant Disk Arrays: Reliable, Parallel Secondary Storage*. PhD thesis, University of California at Berkeley, December 1991.

[9] John H. Hartman, Ian Murdock, and Tammo Spalink. The swarm scalable storage system. Technical Report TR99-06, The Department of Computer Science, University of Arizona, Thursday, April 1 1999. Wed, 08 Jan 99 00:00:00 GMT.

[10] Ian Murdock and John H. Hartman. Swarm: A log-structured storage system for linux. In *Proceedings of the FREENIX Track: 2000 USENIX Annual Technical Conference (FREENIX-00)*, pages 1–10, Berkeley, CA, June 18–23 2000. USENIX Ass.

[11] David A. Patterson, Garth Gibson, and Randy H. Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID). In *Proceedings of the 1988 ACM Conference on Management of Data (SIGMOD)*, pages 109–116, Chicago, IL, June 1988.

[12] Antony Rowstron and Peter Druschel. Storage management and caching in PAST, A large-scale, persistent peer-to-peer storage utility. In Greg Ganger, editor, *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP-01)*, volume 35, 5 of *ACM SIGOPS Operating Systems Review*, pages 188–201, New York, October 21–24 2001. ACM Press.

[13] J. R. Santos and R. Muntz. Performance analysis of the RIO multimedia storage system with heterogeneous disk configuration. In *ACM Multimedia 98*, pages 303–308, Bristol, UK, September 1998. ACM.

[14] Jose Renato Santos and Richard R. Muntz. Comparing random data allocation and data striping in multimedia servers. Technical Report 980038, University of California, Los Angeles, Computer Science Department, November 2, 1998.

[15] Schwabe and Sutherland. Flexible usage of redundancy in disk arrays. *MST: Mathematical Systems Theory*, 32, 1999.

[16] Eric J. Schwabe and Ian M. Sutherland. Flexible usage of parity storage space in disk arrays. In *8th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA '96)*, pages 99–108, New York, USA, June 1996. ACM.

[17] Eric J. Schwabe, Ian M. Sutherland, and Bruce K. Holmer. Evaluating approximately balanced parity-declustered data layouts for disk arrays. In *Proceedings of the Fourth Workshop on Input/Output in Parallel and Distributed Systems*, pages 41–54, Philadelphia, May 1996. ACM Press.

[18] D. P. Siewiorek and R. W. Swarz. *Reliable Computer Systems: Design and Evaluation*. Digital Press, Bedford, Mass., 1992.

[19] Fouad A. Tobagi, Joseph Pang, Randall Baird, and Mark Gang. Streaming RAID: A disk array management system for video files. In *Computer Graphics (Multimedia '93 Proceedings)*, pages 393–400. ACM, Addison-Wesley, August 1993.

[20] H. M. Vin, S. S. Rao, and P. Goyal. Optimizing the placement of multimedia objects on disk arrays. In *International Conference on Multimedia Computing and Systems*. IEEE Computer Society, May 1995.