2016-12-01

A Secure, Reliable and Performance-Enhancing Storage Architecture Integrating Local and Cloud-Based Storage

Christopher Glenn Hansen
Brigham Young University

A Secure, Reliable and Performance-Enhancing Storage Architecture Integrating Local and Cloud-Based Storage

Christopher Glenn Hansen

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Electrical and Computer Engineering

Brigham Young University

ABSTRACT

A Secure, Reliable and Performance-Enhancing Storage Architecture Integrating Local and Cloud-Based Storage

Christopher Glenn Hansen
Department of Electrical and Computer Engineering, BYU
Master of Science

The constant evolution of new varieties of computing systems - cloud computing, mobile devices, and Internet of Things, to name a few - have necessitated a growing need for highly reliable, available, secure, and high-performing storage systems. While CPU performance has typically scaled with Moore's Law, data storage is much less consistent in how quickly performance increases over time. One method of improving storage performance is through the use of special storage architectures. Such architectures often include redundant arrays of independent disks (RAID). RAID provides a meaningful way to increase storage performance on a variety of levels, some higher-performing than others. The fastest performing arrays, however, come at the expense of data reliability, preventing their use in many applications. In this thesis, a storage architecture is presented which utilizes the highest performing RAID architecture (RAID 0) combined with highly available cloud storage in order to maximize storage performance and reliability, while providing an additional benefit of security that is not currently provided by similar storage architectures. Various levels of RAID and the current field of RAID and RAID-like technologies are outlined. Furthermore, the benefits of combining local and cloud storage in RAID configurations, including the ability to recover data in the event of a simultaneous failure of *all* local disks within an array, are discussed. Reliability calculations are performed, and a 3x Mean Time to Failure (MTTF) improvement over existing systems is shown. The MTTF, privacy, read/write throughput, and usable storage percentage improvements of local+cloud-RAID 4 and local+cloud-RAID-DP are discussed. Thereafter, a software implementation of local+cloud-RAID 4 is presented and discussed, allowing for notable benefits over existing architectures in terms of reliability and privacy while maintaining performance. Finally, a novel hardware-based local+cloud-RAID controller architecture is described and discussed.

Keywords: RAID, storage, security, privacy, reliability, architecture

ACKNOWLEDGMENTS

I would like to thank Dr. James Archibald for his willingness to take me on as an advisee, and his assistance in editing my papers. Thank you to the Embedded Security Group at BYU for providing a forum for discussion and idea generation. Thank you to Dr. Michael Wirthlin for teaching the skills required for reliability analysis. Thank you to Dr. Doran Wilde for his guidance on this thesis. Thank you to all of the professors at BYU who have contributed to my education. Finally, I thank my family and friends who have supported me in many ways.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

As computing systems have become more ubiquitous within every facet of daily life, the importance for these systems to work reliably and securely ever increases. Computers are used for many critical applications in virtually all economic sectors. System failures, including permanent loss or unavailability of data, can be catastrophic. Long-term data reliability has become increasingly important as both governments and corporations move their sensitive information to digital storage media, seeking permanent solutions. With the transference of secure systems to the cloud, and with the advent of secure online systems such as online banking, medical computing, government services, and enterprise systems, the need for high availability, reliability, and security within online storage has necessarily increased. These needs have been extensively discussed [2–7].

The need for online storage has become increasingly pronounced as devices which rely on cloud storage as their primary nonvolatile storage mechanism gain in popularity. Instances of such devices range from Google's Chromebooks to various Internet-connected IP cameras. The Internet of Things (IoT) presents another use case for cloud storage. Because many IoT-connected devices, by nature, are designed to be small and inconspicuous, they often lack the physical space for large amounts of storage. Developing novel storage architectures to support the varied uses of the IoT is an active research area [8–11]. While these systems often rely heavily on the cloud for needed data storage, cloud computing is not without its disadvantages. Data accesses to the cloud can be slow, negatively impacting performance. Additionally, utilizing a single cloud provider as a complete storage solution carries with it the inherent risk of cloud server downtime, causing data unavailability. Furthermore, when user data is exclusively stored on servers controlled by a single provider, data privacy and secrecy are not necessarily maintained. A compromised or rogue service provider, or one coerced by a government to disclose server data, may access, expose, or modify user data. These concerns are widely known [4, 6, 7, 12–19].

Redundant arrays of independent disks, or RAID, provide redundancy and/or performance-enhancing benefits which can be applied to local and cloud-based storage. Various levels of RAID exist, each with its own distinct benefits to reliability and performance. With the exception of RAID 0, RAID provides redundancy which improves data reliability. Furthermore, most RAID levels improve performance, especially read speeds.

The importance of system performance, efficiency, security, data reliability and availability have only grown over the decades, and storage efficiency has become increasingly more important as cloud storage providers seek to offset operation costs while maximizing storage capabilities for customers. Efficiency is defined as the percentage of storage overhead incurred by various RAID configurations, with parity generation being more efficient than mirroring. Three metrics govern the choice of which RAID level is chosen: performance, reliability, and feasibility. The metric of feasibility includes efficiency, as well as cost and hardware requirements, i.e. how many drives are necessary to implement the selected RAID level, and what controller technology is available.

With the steady migration of data to the cloud, the importance of maintaining data integrity, privacy, and reliability remains. It follows that RAID, while having been extensively used and analyzed in configurations including only local drives, now has application to cloud drives. Various schemes have been presented which utilize multiple cloud drives to increase the security of data stored in the cloud. Some of these schemes use RAID-like methods to improve cloud reliability, security, and performance [2, 4, 5, 7, 12–14, 20–22]. Schnjakin et al. describe a method of creating a RAID 0 array which includes only cloud storage, and an evaluation of their system shows how RAID 0 across cloud media increases data security and general performance, although server downtime presents a problem for availability [23, 24]. Other schemes aim to increase data reliability and availability in the cloud through redundancy, workload analysis, and other techniques [4, 15, 16, 18, 19, 25, 26]. Schemes which implement dynamic selection of cloud-based providers have also been implemented. When properly designed, any system can detect misbehaving cloud storage providers and take corrective action, migrating its data if needed [13,27,28]. The extent of this work is such that it has necessitated a common taxonomy, with terminology including “aggregated clouds”, “bursted clouds”, “cloud federations”, and “distributed clouds”. [29].

These schemes effectively implement various RAID levels within cloud storage media. However, the inclusion of storage devices local to a computer within the hitherto proposed and analyzed schemes is a conspicuous gap in research. In this paper, we discuss further extending the methods previously proposed to include combinations of both local and cloud drives, and we analyze the impact of various configurations of such a scheme. Appropriate RAID level selection for a given application is dependent on a number of factors, including whether a system utilizes local storage, cloud storage, or a combination of both. Additionally, system designers must balance cost with the desire to improve the storage parameters of performance, security, and reliability. These parameters and compromises will be discussed for each configuration presented.

Herein, RAID levels and RAID-like technologies are discussed in order to provide a foundation for expanding the current field of storage architectures. A novel storage architecture which can bolster a system's storage performance, reliability, and security is proposed. This architecture is based on combining RAID 4 with local and cloud storage. The performance of a software implementation of this architecture is analyzed and compared to many other software RAID configurations. Additionally, a hardware implementation is proposed and described.

The contributions of this thesis include the following:

1. a survey of various existing RAID levels and technologies.
2. a novel local and cloud-based storage architecture derived from RAID 4 which can bolster storage and system performance, reliability, and security.
3. an analysis of the read and write throughput performance of a software-based implementation of a proposed architecture, and a comparison to other RAID configurations.
4. a proposal for a hardware-based implementation of local+cloud-RAID 4 which combines a microprocessor, custom RAID controller logic, and cloud APIs.

Chapter 2 presents an overview of historical RAID levels and current RAID technologies. Chapter 3 presents the proposed architecture. In Chapter 4, a software-based implementation and proof of concept is described and analyzed. In Chapter 5, a hardware-based architecture to extend and improve the performance of the software-based implementation is presented. In Chapter 6, conclusions are drawn and future work based on the proposed architecture is discussed.

CHAPTER 2. RAID AND RAID-LIKE TECHNOLOGIES

This chapter surveys the historically standard RAID levels, nested RAID levels, open-source and free RAID-like systems, non-standard RAID levels, and proprietary RAID and RAID-like systems. Different RAID levels have various implications for system availability, reliability, performance, and sometimes, security, as discussed below. Also, in this chapter a reliability analysis of RAID is performed, which utilizes Markov models.

Often enhancements to reliability and/or performance provided by a specific level come at the expense of another desirable metric, such as efficiency. When reliability and security can be improved without negatively affecting system performance, such enhancements become immensely more attractive to designers and architects. In Chapter 3, an architecture that can improve reliability, availability, and security, while maintaining the performance of faster RAID levels, is proposed and discussed.

2.1 Overview of Basic RAID Configurations

This section provides an overview of the basic RAID levels in existence on which most other RAID technologies are based. Patterson et al. outlined these RAID levels in their groundbreaking paper in 1988 [1].

2.1.1 RAID 0

RAID 0 is defined as the striping of data between two or more drives, and is depicted in Figure 2.1. In this configuration, data is striped at the block level across multiple disk drives. Thus, only a portion of the data is stored on each drive; every disk must function properly in order to have the complete set of working data. Thus, if any disk fails, the original data is permanently lost.

In general, this configuration greatly enhances the performance of reads and writes, while sacrificing reliability. In fact, the reliability of RAID 0 is *always* lower than the lowest individual

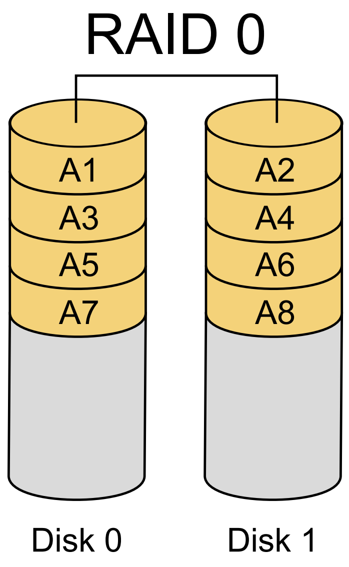

Figure 2.1: RAID 0, showing block-level striping of data across two disks. This work was created by Cburnett and is licensed under CC BY-SA 3.0 [30].

reliability of the drives included in the array. Additionally, because all drives are included in servicing *any* read or write request, drives are subject to an increased level of stress, further diminishing reliability. Without any redundancy to mitigate detrimental effects, reliability decreases. Thus, this configuration is not suitable for important, sensitive, or mission-critical data. However, this configuration is popular among enthusiasts who prioritize performance over reliability, especially in the case where the loss of data does not have a significant detrimental impact. On the other hand, with the increasing popularity of software-based cloud backups, risk can be substantially mitigated, even with RAID 0. Finally, RAID 0 is easy to implement as it requires a minimum of just two drives in order to function. If data reliability is not of primary concern, then RAID level 0 is a feasible option for the performance-minded, with vastly increased drive performance and low hardware requirements [31–33]. This RAID level is common in practice.

2.1.2 RAID 1

Raid 1 is also common in practice, and is depicted in Figure 2.2. It consists of the mirroring of a drive to one or more other drives. Thus, all drives contain identical data. This can be effectively categorized as a backup of a drive to one or more other drives. This RAID level

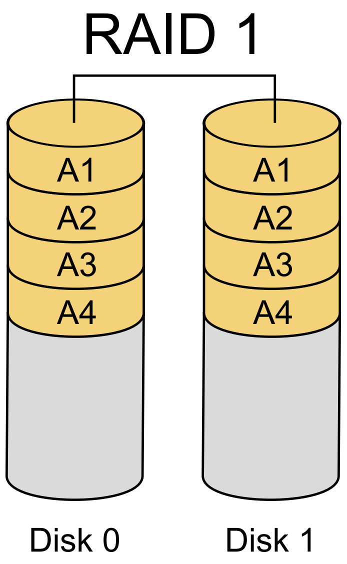

Figure 2.2: RAID 1, showing block-level mirroring of data across two disks. This work was created by Cburnett and is licensed under CC BY-SA 3.0 [30].

Table 2.1: Example parity calculations

	Disk 1	**Disk 2**	**Disk 3**	**Odd Parity**	**Even Parity**
Data 1	0	0	1	0	1
Data 2	1	1	0	1	0
Data 3	0	0	0	1	0
Data 4	1	1	1	0	1

offers high reliability and access performance equal to a non-RAID system, but it incurs increased system cost and overhead, as well as decreased storage efficiency. However, the system is easy to implement and requires a minimum of just two drives. Feasibility depends on system constraints for cost, efficiency, and complexity, but RAID 1 is an attractive option for cost-constrained applications requiring high reliability [31, 33, 34]. However, security may actually be *decreased* with the addition of drives to a RAID 1 array, especially if a mirrored drive is offsite or in an insecure location. This is because the ability to secure multiple locations at a fixed cost decreases as the number of locations where data is stored increases.

2.1.3 RAID 2 and RAID 3

RAID 2 and RAID 3 both involve a combination of striping and parity, incorporating the benefits of RAID 0's striping and RAID 1's redundancy. Table 2.1 shows examples of odd and even parity calculations. Odd parity implements an XOR operation by counting the number of 1 bits within a set of data and outputting a 0 if the data contains an odd number of 1 bits, or outputting a 1 if the data contains an even number of 1 bits. Similarly, even parity implements an XNOR operation by counting the number of 1 bits and outputting a 0 if the data contains an even number of 1 bits, or outputting a 1 if the data contains an odd number of 1 bits.

The result of the XOR or XNOR operation creates redundant data that can be used to recover information in the case of a disk failure. For example, if parity has been stored, and then disk 1 suddenly fails, the original data from disk 1 can be restored by again performing the respective XOR or XNOR on the remaining data and parity.

RAID 2 is depicted in Figure 2.3. RAID 2 stripes data at the bit level. It utilizes a distributed parity scheme calculated with Hamming codes. RAID 3 is depicted in Figure 2.4. This level stripes data at the byte level, and utilizes odd parity and a dedicated parity disk, unlike RAID 2. Both levels require a large performance overhead for computation of redundant data because data is striped on such small levels. Synchronization of multiple drives is difficult when reading and writing at the bit and byte levels; all drives must have the exact same rotational speeds in order to remain synchronized. Such a requirement is often infeasible because hard disk drives suffer from difficulty in synchronizing rotational speeds. However, solid-state drives do not have this drawback. An additional drawback to levels 2 and 3 is their stunted performance in degraded mode, which occurs when one of the drives in the array fails and the array is being reconstructed. While theoretical reliability is good for these levels, performance greatly suffers in practice, rendering these levels largely obsolete, regardless of feasibility. Additionally, security is not meaningfully impacted through the implementation of these levels. These drawbacks have precluded the widespread adoption of both RAID 2 and RAID 3. Because RAID 5 offers superior performance and equal reliability, RAID 2 and RAID 3 are rarely used in practice [31, 33, 35].

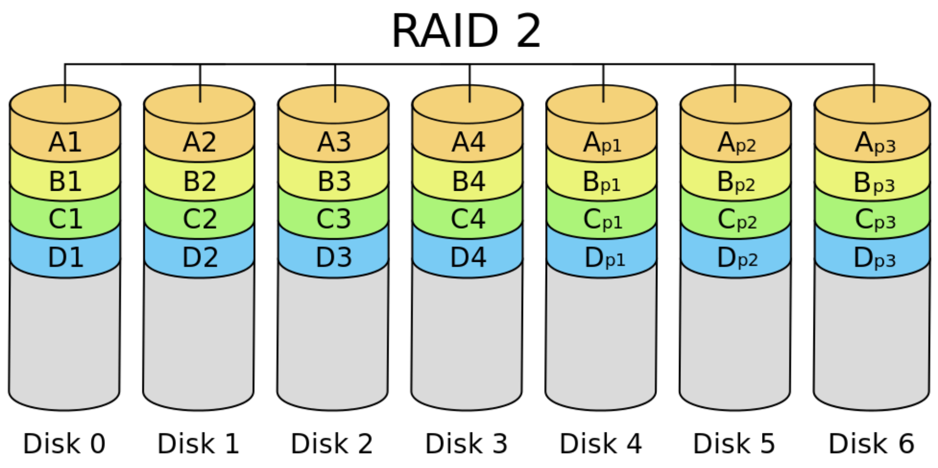

Figure 2.3: RAID 2, showing striping and parity at the bit level. This work was created by Cburnett and is licensed under CC BY-SA 3.0 [30].

2.1.4 RAID 4

RAID 4 is much like RAID 3. RAID 4 is depicted in Figure 2.5. This level utilizes a dedicated parity disk, but stripes data on the level of blocks rather than bytes. Because of the need for a dedicated parity disk, write performance suffers because writes to the parity disk become a bottleneck, just as for RAID 3. However, this level improves upon RAID 3 by effectuating striping on the block-level. Fewer parity bits are required for RAID 4 than for RAID 3, thus mitigating somewhat the bottleneck of writing parity bits to the dedicated parity drive. This level of RAID offers good reliability, but severely decreased write performance, as the dedicated parity drive bottlenecks the system when frequent parity calculations are required. Additionally the parity drive undergoes much more stress than the other drives in the array due to more frequent accesses. This can lead to quicker failures of that drive, decreasing reliability. This level of RAID requires a minimum of three drives [31, 33, 35]. With the performance penalty of a dedicated parity drive, and with the relative advantages of RAID 5, this level of RAID is not currently used in practice. However, it will be further reexamined in later sections and shown to have advantages over RAID 5 and RAID 6 within arrays that include cloud storage.

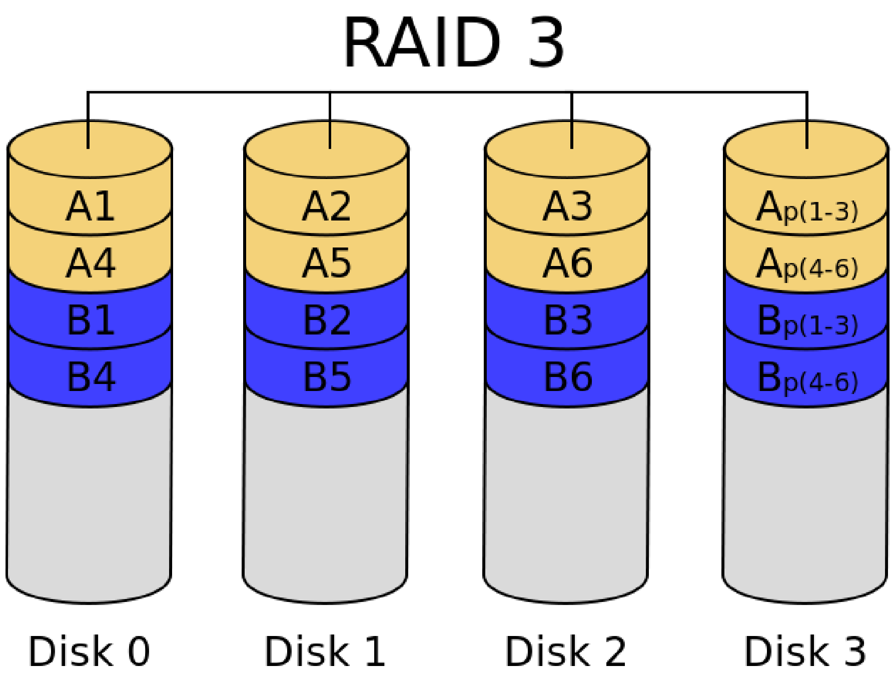

Figure 2.4: RAID 3, showing striping and parity at the byte level. This work was created by Cburnett and is licensed under CC BY-SA 3.0 [30].

2.1.5 RAID 5

RAID 5, depicted in Figure 2.6, is similar to RAID 4, but is traditionally regarded as superior. RAID 5 does block-level striping and parity calculation, but distributes the storage of parity bits among all drives in the array. Thus, no one drive is dedicated to absorbing the extra overhead of parity writes. As Figure 2.6 shows, RAID 5 distributes the stress and overhead of parity evenly among all drives in the array, eliminating the bottleneck of a dedicated parity drive experienced by RAID 4. RAID 5 has been commonly used in server environments and cloud storage because it offers better reliability than both simplex systems (no RAID) and RAID 0, better storage efficiency and performance than RAID 1, and better write performance and reliability than RAID 4. RAID 5 requires a minimum of three drives to function [31, 33, 36].

2.1.6 RAID 6

RAID 6, which is depicted in Figure 2.7, is similar to RAID 5. However, it also includes the addition of a second layer of parity and the ability to withstand two separate drive failures. The

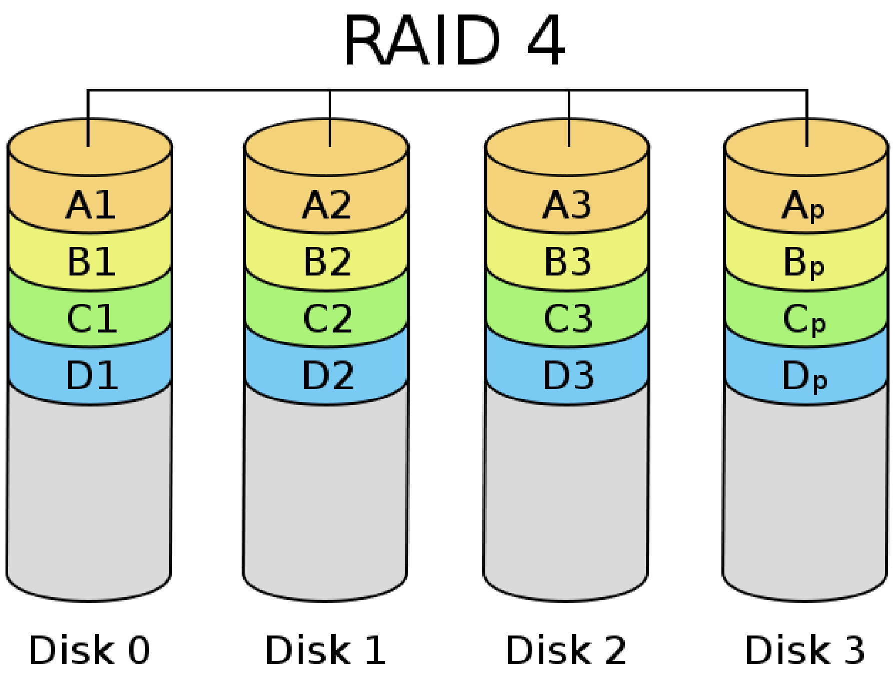

Figure 2.5: RAID 4, showing striping and dedicated parity disk. This work was created by Cburnett and is licensed under CC BY-SA 3.0 [30].

figure depicts two orthogonal layers of distributed parity, labeled p and q. RAID 6 becomes more attractive as the number of drives inside a RAID array increases. For instance, in an array of 4 drives, single level parity, which can tolerate one drive failure, may be sufficient. However, if one desires to construct a larger array, failures become more frequent, and the ability to tolerate two drive failures may be desired in order to maintain an available system. While RAID 6 improves up the reliability of RAID 5, it incurs higher cost, higher write overhead, and an efficiency penalty. However, read performance among the arrays can remain constant. This is an attractive RAID level for large cloud data storage centers and server farms, and it is common in practice [31, 33, 36].

The parity generation scheme in RAID 6 is different than that of RAID 5. RAID 5 and the first parity layer of RAID 6 utilize a simple XOR operation of all of the data in each row of the array. RAID 6, in order to calculate a second, orthogonal layer of parity for tolerance of a loss of a second disk, must utilize a separate method for parity calculation for the second layer of parity. This method utilizes Galois fields and is described by Anvin and published on the Linux kernel site kernel.org with examples [37].

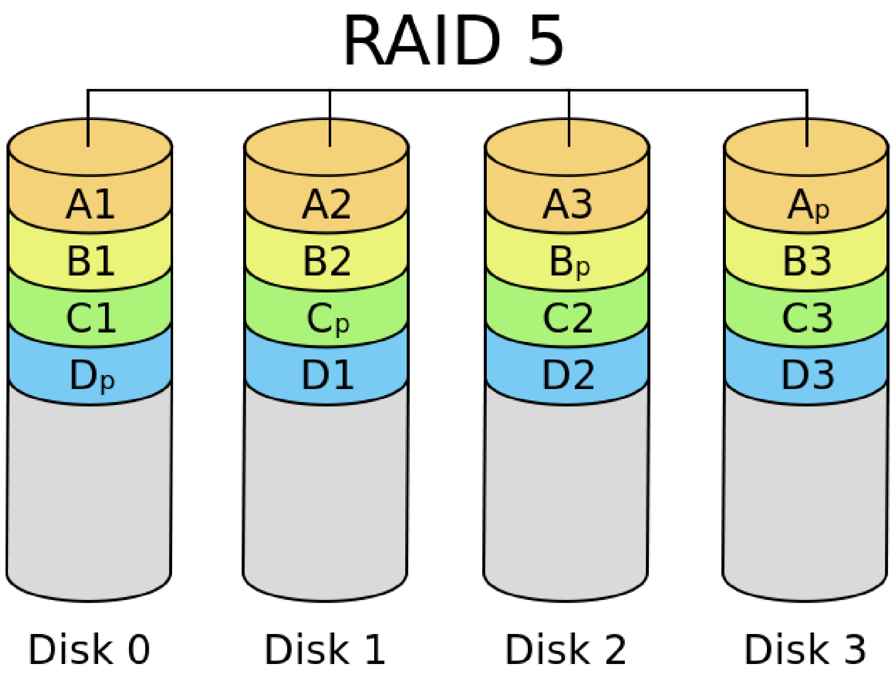

Figure 2.6: RAID 5, showing striping and distributed parity. This work was created by Cburnett and is licensed under CC BY-SA 3.0 [30].

2.2 Higher-Order RAID Configurations

Higher-order RAID configurations, also known as nested RAID configurations, combine multiple RAID levels within a single array. Theoretically, any RAID levels can be combined with any other level. However, not all of these levels are meaningful to improve system functionality. Outlined below are some meaningful configurations which are used.

2.2.1 RAID 10

RAID 10, shown in Figure 2.8, depicts a higher level of striping along with a lower level of mirroring. This is one of the most basic hybrid or nested-RAID arrays. It offers great increases in both performance and reliability. Read performance and write performance both exceed that of RAID 5 and 6, and RAID 10 can tolerate two drive failures, as long as those failures are not in the same RAID 1 array. Otherwise, it always tolerates the failure of one drive. However, drive space efficiency is only 50 percent in a RAID 10 array, whereas RAID 5 or 6 arrays allow for upwards of 90% storage efficiency, with a minimum of 67% efficiency. Parity does not have to be calculated

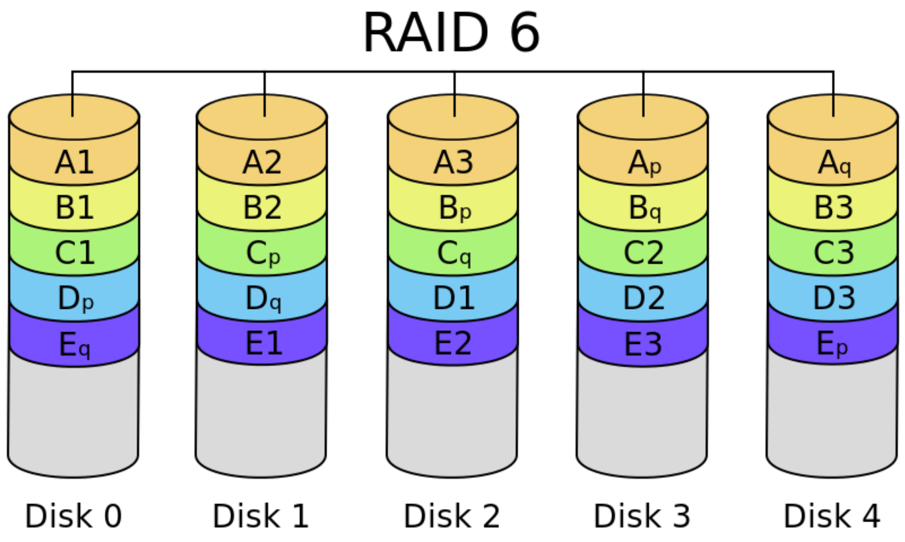

Figure 2.7: RAID 6, showing striping and two layers of distributed parity. This work was created by Cburnett and is licensed under CC BY-SA 3.0 [30].

for RAID 10, allowing for faster write speeds. Furthermore, seek times are generally faster for RAID 10 than for RAID 5 and 6, due to the absence of parity bits, allowing for faster reading. This RAID level requires a minimum of four drives, and has become common for high performance, high reliability systems [31, 34].

2.2.2 RAID 01

RAID 01, shown in Figure 2.9, consists of a mirror of stripes. This is another nested RAID configuration, and is equal to RAID 10 in performance and efficiency. However, the reliability of RAID 01 is lower than that of RAID 10. This is due to the fact that RAID 10 not only tolerates the loss of one drive, but can also tolerate the case of two drive failures in different RAID 1 arrays. RAID 01 can tolerate the loss of just one drive; with the loss of any drive beyond the first it is unable to reconstruct the array after failure. For this reason, RAID 01 is not often used in practice, since it is entirely supplanted by RAID 10 in reliability while offering no better performance [31, 33, 34].

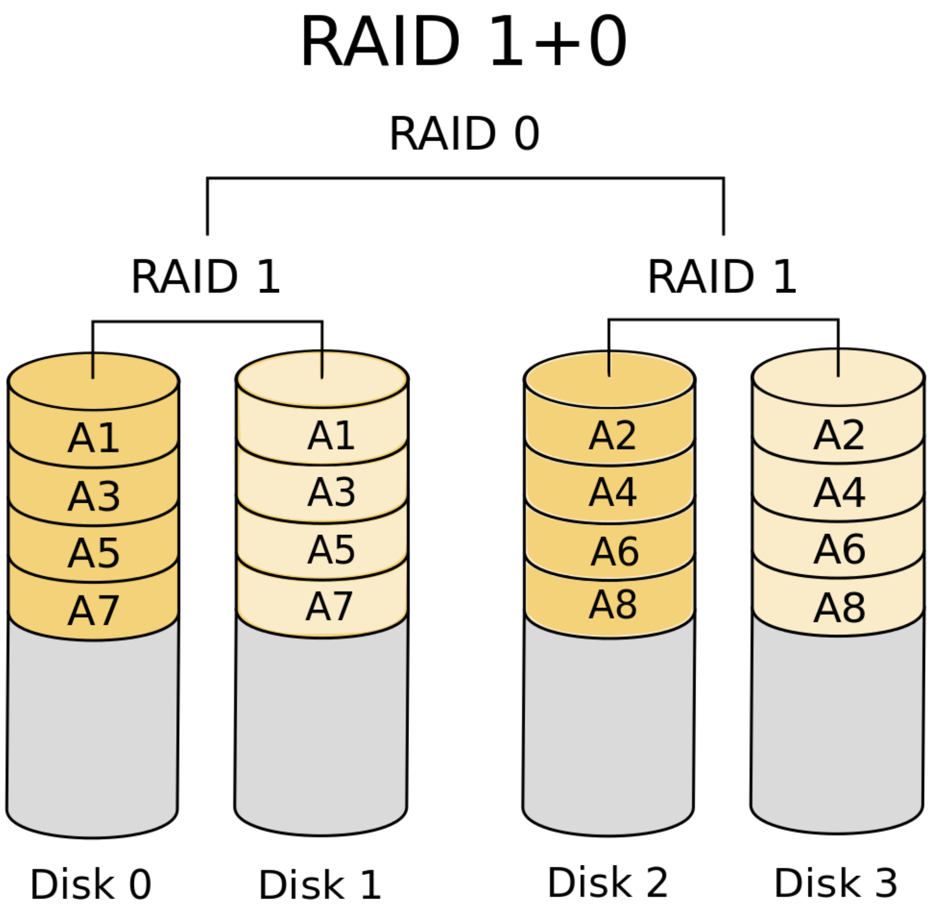

Figure 2.8: RAID 10, showing a stripe of mirrors configuration. This work was created by Nudel-Suppe and is licensed under CC BY 3.0 [30].

2.2.3 RAID 50

RAID 50 is shown in Figure 2.10. This figure depicts a stripe of RAID 5 subarrays, significantly improving upon both the write performance and reliability of RAID 5. One drive from each subarray can fail without the total loss of array data [30, 33]. This configuration requires a minimum of six disks to operate. It offers improved read and write speeds over RAID 5 at the expense of increased complexity.

2.2.4 RAID 60

RAID 60, shown in Figure 2.11, significantly improves upon the reliability of RAID 50, RAID 5, and RAID 6. RAID 6 has significantly higher reliability than RAID 5, so it naturally

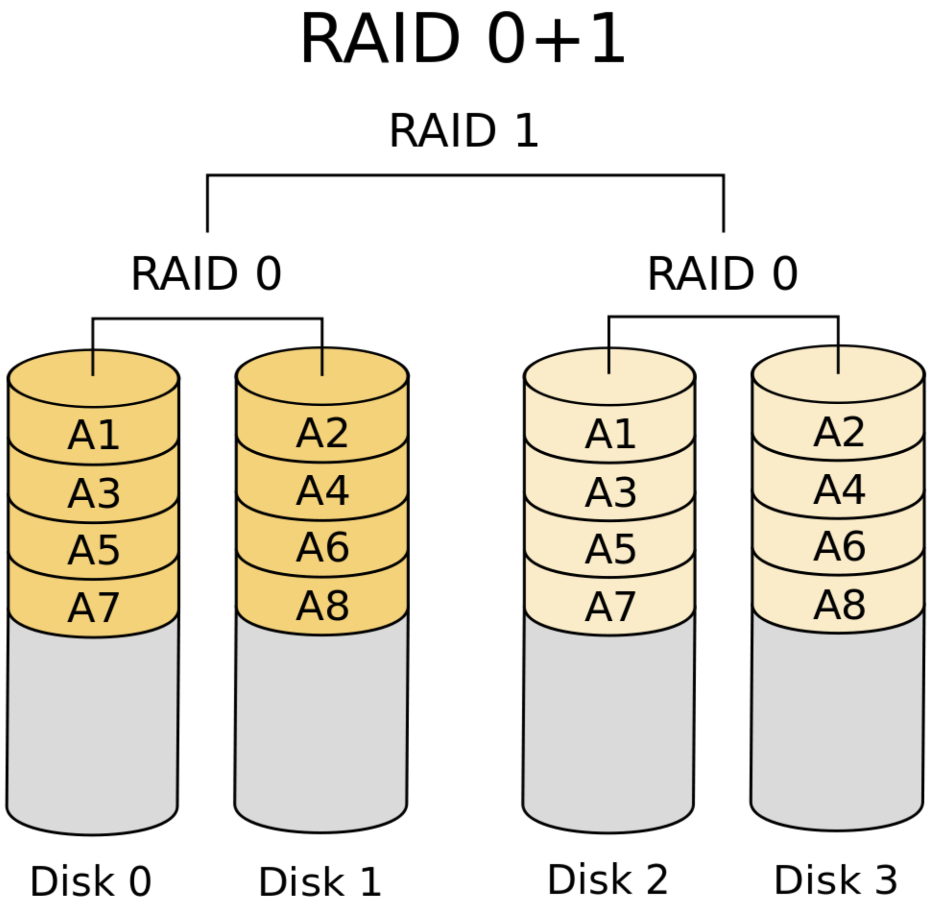

Figure 2.9: RAID 01, showing a mirror of stripes configuration. This work was created by Wheart and is licensed under CC BY-SA 3.0 [38].

follows that RAID 60 also has much greater fault tolerance than RAID 50. RAID 60 can tolerate the loss of up to two drives within each RAID 6 subarray, which would be a massively simultaneous loss of data [33]. In Figure 2.11, the system can recover from up to 50% of drives failing. Performance is inferior to RAID 50, as RAID 60 incurs more overhead than RAID 50 due to extra layers of distributed parity.

2.2.5 RAID 100

RAID 100, shown in Figure 2.12, consists of a stripe of RAID 10 subarrays, which improves both upon the performance and reliability of RAID 10. Like RAID 10, it is a non-parity array, thus requiring less computational overhead than arrays that require parity generation [39].

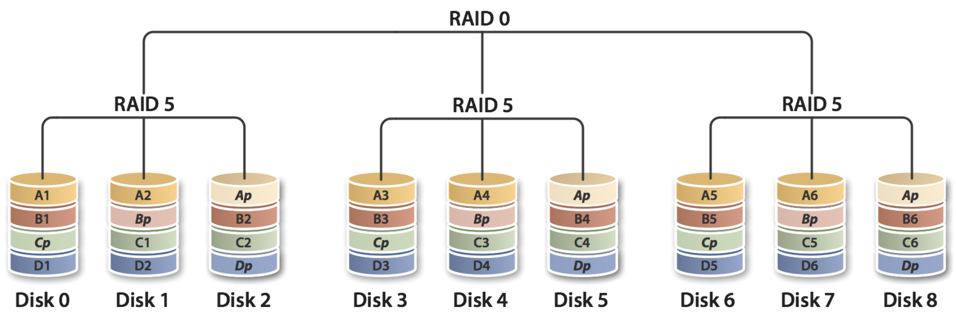

Figure 2.10: RAID 50, showing a striping of RAID 5 subarrays. This work was created by Kauberry, is cropped from the original, and is licensed under CC BY-SA 3.0 [30].

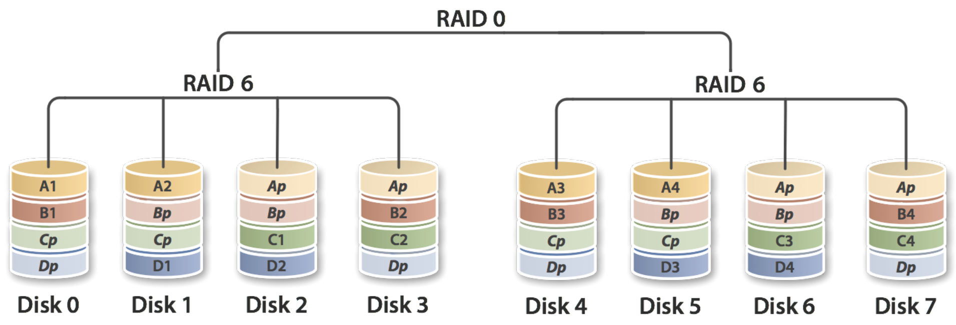

Figure 2.11: RAID 60, showing a striping of RAID 6 subarrays. This work was created by Rusl and is licensed under CC BY-SA 3.0 [38].

This level can also tolerate up to 50% of drives failing, and offers the fastest performance of any hitherto discussed RAID configuration, at the expense of requiring the most number of disks and incurring 50% storage space overhead.

2.2.6 More Higher-Order Nested Levels

RAID levels can be combined at will. For example, RAID 61 is created by mirroring two RAID 6 arrays, and RAID 16 is created by creating a RAID 6 array with multiple RAID 1 arrays [40].

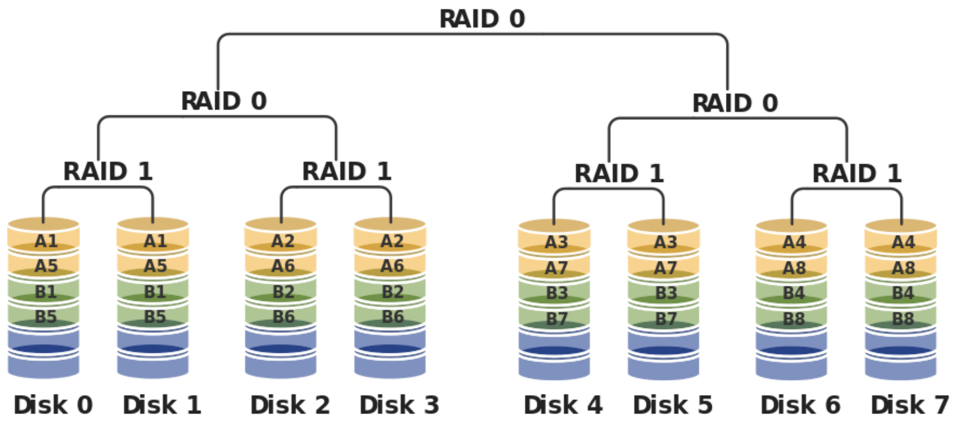

Figure 2.12: RAID 100, showing a striping of RAID-10 subarrays. This work was created by Kauberry, is cropped from the original, and is licensed under CC BY-SA 3.0 [38].

2.3 Open-Source/Free RAID-like Systems

This section presents storage systems and frameworks that are free or open-source, many of which are widely used in various applications in order to bolster storage availability, reliability, and performance.

2.3.1 Apache Hadoop

Apache Hadoop is an open-source software framework that aims to solve problems in distributed computing and storage, including performance optimization and fault tolerance. The framework allows Hadoop to break up workloads into small batches across multiple machines, and does so with redundancy. Hadoop handles hardware failures at all levels. Hadoop has a 13 year history, originating with the Google File System in a paper in 2003, and is used by many prominent technology and media companies, including Amazon, Adobe, Alibaba, Facebook, Google, Hulu, IBM, Last.fm, LinkedIn, Rakuten, Sling, Spotify, Twitter, Yahoo, and numerous universities [41, 42].

2.3.2 BeeGFS

BeeGFS, which was formerly known as FhGFS, is a free parallel cluster file system designed to provide flexibility, scalability, high throughput, and ease of use. However, reliability is not a primary concern of BeeGFS. BeeGFS stripes data across multiple servers which allows for very fast simultaneous accesses to multiple distinct locations, effectively combining the throughput of all remote servers [43]. This framework effectively implements a RAID 0 array, with each "disk" consisting of a remote server.

2.3.3 MooseFS

MooseFS is an open-source distributed filesystem which was developed by Core Technology. The primary market for MooseFS is data centers, as it aims to provide high availability, high reliability, high performance, and high scalability. Other useful enterprise features include load balancing, security, snapshots, and quotas [44].

2.3.4 GlusterFS

GlusterFS was developed by Gluster, Incbut is now owned by Red Hat. It is open-source. GlusterFS has applications in media servers, content delivery networks (CDNs), and cloud computing. The premise is to collect remote storage servers and combine them together into a single entity [45].

2.3.5 Btrfs

The B-Tree File System, or btrfs, is an GPL licensed filesystem for Linux, which was developed by multiple companies, including Facebook, Fujitsu, Fusion-IO, Intel, Linux Foundation, Netgear, Novell, Oracle, and Red Hat [46]. Btrfs is designed to enable fault tolerance and easy repair and administration of drives. It implements a copy-on-write (CoW) system which enables easy drive mirroring and shadowing. Furthermore, btrfs supports snapshots, compression, checksums, RAID 0, RAID 1, some higher RAID levels, incremental backups, file system checking, defragmentation, and many other useful features which were previously lacking in Linux [47].

2.3.6 ZFS

Oracle's ZFS was designed to support and enable high data capacities, fault tolerance, compression, snapshots, and many other useful filesystem features. Originally it was developed by Sun and their implementation remained open until 2010. The same year, Illumos was founded to continue the open-source work on ZFS implementations, and in 2013 the OpenZFS project began [48].

FreeNAS, a popular storage operating system, utilizes ZFS as its filesystem. The project cites the open-source nature of ZFS, as well as its “unprecedented flexibility and an uncompromising commitment to data integrity” [49].

2.3.7 Linux mdadm

The open-source Linux mdadm (multiple device administration) utility was developed by Neil Brown of SUSE and released in 2001 [50]. It provides a lightweight tool to create and manage RAID volumes on an individual system. The mdadm utility is the de facto default for users who want to implement RAID on a Linux system. The utility supports RAID 0, RAID 1, RAID 4, RAID 5, RAID 6, and RAID 10 [51].

In Chapter 3 of this thesis, a new RAID architecture based on RAID 4 is proposed. Using mdadm, an instance of this architecture is presented and discussed, and in Chapter 4, performance measurements of various RAID implementations of mdadm running on external USB flash drives are presented.

2.4 Non-standard RAID Levels

In addition to the standard RAID levels described above, many vendors implement other, non-standard, levels. Sometimes these levels are proprietary, or sometimes they have the same functionality but have different vendor-specific names. This section describes many instances of non-standard RAID levels that exist.

2.4.1 JBOD

JBOD (Just a Bunch of Disks) is not a level of RAID, but should be mentioned for the sake of completeness. JBOD treats a conglomerate of disks as a single entity, regardless of size, manufacturer, or any other disk parameters. When users do not care about the reliability of a set of disks and only need to aggregate a collection of mismatched disks to attain a single entity with maximum size, JBOD is an attractive option. Such a use case might be a scratch drive where temporary files are stored, like a swap space, or a JBOD that is mirrored elsewhere to make it more reliable.

2.4.2 Drive Spanning

Drive spanning is often used in conjunction with JBOD. Drive spanning entails concatenating drives together so that when one drive fills up, the filesystem spills over seamlessly into the next drive in the array. This is also called BIG or SPAN.

2.4.3 MAID

Massive arrays of idle disks (MAID) is used for nearline storage. Nearline storage is neither online nor offline storage. Online storage is defined as readily available storage to a system, such as a spinning hard disk. Offline storage is long-term, archival storage, such as a tape backup. Nearline storage is storage that is between the two, which trades increased latency for an increased amount of storage and redundancy. These disks are used when offline storage is too cumbersome and when an online storage architecture is not appropriate. MAID is similar to RAID, and may implement RAID as well [52]. An example of MAID is a backup system which is accessed periodically and used for scheduled periodic backups of a database.

2.4.4 RAID 1E

Some RAID 1 implementations, when utilized with more than two disks, create instead an array which combines striping and mirroring, much like RAID 10, but is a distinct configuration. While RAID 10 requires a minimum of four disks, and always requires an even number of disks, RAID 1E requires at least three disks, and the total number can be even or odd. This level works by first striping all array data across all disks. Then, a mirror is created and striped across all the disks once again, although offset from the original stripe. RAID 1E does not work with two disks because mirroring and striping cannot be combined with only two disks. RAID 1E creates a high level of performance while maintaining redundancy [53].

RAID 1E is an instance of a fairly common configuration that has different names among different industry vendors, including “striped mirroring”, “enhanced mirroring”, or “hybrid mirroring” [53].

2.4.5 ZFS RAID-Z

ZFS additionally provides a non-standard RAID level called RAID-Z, which, like RAID 5, combines striping and parity across multiple disks. However, through tightly integrating the filesystem and the physical array, RAID-Z is able to improve upon RAID 5 by relaxing constraints for stripe width. Additionally, RAID-Z can fix corrupted data in-flight when it detects blocks that fail the block checksum. RAID-Z provides three options which correspond to the number of levels of parity contained within the array, and therefore the number of disk failures that can be tolerated in the array. These three options are RAID-Z1, RAID-Z2, and RAID-Z3 [54].

2.4.6 Drive Extender

Drive Extender was a utility built into Windows Home Server which effectively allowed a user to group together any number of drives arbitrarily to act as a single pool, much like JBOD and SPAN. Additionally, Drive Extender provided a level of redundancy similar to RAID 1, enabling fault tolerance without the use of any specialized hardware [55].

Drive Extender was officially discontinued by Microsoft in 2011. Third party utilities, such as DriveBender by Division M and DrivePool by StableBit, have come to fill the gap by providing similar functionality [56, 57].

2.5 Proprietary RAID Levels and RAID-like Technologies

Many RAID and RAID-like technologies exist which are not open standards. Various companies have implemented their own proprietary RAID solutions in order to improve reliability and performance. Security is not a primary focus for most of these configurations. This section presents an overview of proprietary RAID solutions.

2.5.1 Intel Rapid Storage Technology

Intel Rapid Storage Technology allows various volumes or partitions on a drive to be assigned to different RAID arrays. It supports RAID 0, 1, 5, and 10. It is a firmware implementation, rather than a higher-level software or lower-level hardware implementation. Intel Matrix RAID allows creation of RAID volumes from the BIOS, easily enabling common configurations such as striping an OS across two disks and mirroring a larger disk used for data [58].

2.5.2 IBM Spectrum / IBM GPFS

IBM Spectrum aims to help other businesses reduce costs, scale operations, and manage data up to very large scales. The primary goals of IBM Spectrum Scale is to provide extreme scalability, data analytics tools, and automated cost optimization through selection of the most cost effective storage devices [59].

2.5.3 NetApp RAID-DP

RAID-DP (double-parity), developed by NetApp, is an extension of RAID 4. Like RAID 4, it utilizes dedicated parity drives. However, unlike RAID 4, it utilizes a second dedicated parity drive with a second level of parity, much like RAID 6. However, it differs from RAID 6 in that RAID 6 utilizes distributed parity and Galois Fields to calculate the second parity layer. RAID-DP utilizes a proprietary method for second-layer parity generation by calculating both row and diagonal parity across the array.

This technology is comparable to the proposed scheme in Chapter 3, where using multiple dedicated cloud parity drives is discussed. The key difference between NetApp RAID-DP and the approach in Chapter 3 is the location of the dedicated parity drives. RAID-DP utilizes local storage, while the new architecture proposes cloud storage for dedicated parity.

2.5.4 unRAID

Lime Technology's unRAID technology combines JBOD with a dedicated parity disk and write caching, enabling quick read and write performance while providing redundancy. This product is popular for media applications and marketed toward home media servers and is free to use for up to three hard disks [60].

In chapter 5, I build upon this scheme of using a dedicated parity disk in conjunction with a fast cache and embedded processor, combining it with cloud APIs for remote storage.

2.5.5 BeyondRAID

BeyondRAID is a technology marketed by Drobo and used on their Network Attached Storage (NAS) products which provides improved data performance and redundancy, but doesn't implement any traditional RAID system. Rather, BeyondRAID aims to automatically choose the appropriate RAID level for given circumstances. It does not require drives to be of equal size. Furthermore, it enables easy array reconfiguration, including expansion, hot sparing, drive reordering, and even changing of RAID levels without data loss [61].

2.5.6 RAID 7 and Storage Computer Corporation

RAID 7 is a technology that was pushed by the now defunct Computer Storage Corporation. It combined striping with dedicated parity and a write cache, like the write cache used in unRAID, and an embedded processor to asynchronously manage parity [62]. This scheme is similar to the appropach proposed in Chapter 3 of this thesis, except dedicated parity resides in the cloud rather than locally, providing many benefits such as increased reliability, security, and reduced complexity for local arrays.

2.6 RAID Reliability Analysis

In this section, a mathematical analysis of the reliability of various RAID levels is presented. Generic reliability equations are formulated with the aid of Markov models representing each RAID configuration. Shooman presents a model-based analysis flow to solve for the reliability of systems such as RAID [31]. This section analyzes RAID arrays that only include local storage disks. This analysis is extended in chapter 3. These analyses are published in the proceedings of ARES 2016 [63].

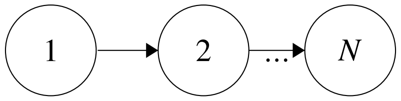

Figure 2.13: A serial reliability model for RAID 0 with N elements

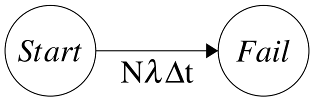

Figure 2.14: A Markov failure model of RAID 0 with N striped drives and constant failure rate λ.

2.6.1 Failure Model

Various failure modes or degraded modes of operation exist within data storage devices. Typically, devices constantly perform self-monitoring and error correction in order to operate reliably. However, when devices have failed to the point that error correcting and various fault-catching techniques no longer allow for reliable operation, and data becomes permanently lost, a drive is said to have failed. The permanent failure mode is the mode of primary interest of this thesis and the ensuing analysis.

2.6.2 RAID 0

Figure 2.13 depicts a serial reliability model for RAID 0 for N disks wherein any state other than state 1 represents failure. Arrows represent a disk failure. The leftmost circle represents a fully-functional array. A serial model implies that no redundancy exists within the system; parallel models include redundancy. All implementations of RAID 0 can be modeled serially, rather than with a parallel model.

Figure 2.14 is a Markov model of the reliability for RAID 0. Markov models and their associated theory is presented by Shooman [31]. In Figure 2.14, N is the number of distinct drives in the RAID 0 array, and λ is the constant failure rate. Equivalent and constant failure rates for all drives are assumed. Thus, the overall failure rate for the array is $N\lambda$. The reliability of this configuration is proportional to the number of drives over which RAID is configured.

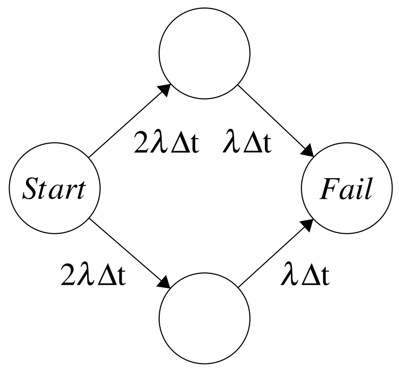

Figure 2.15: A parallel reliability model for RAID 1 with two local drives

Serial reliability behavior is described in the reliability equation presented by Shooman [31]:

$$R(t) = P(x_1)P(x_2)...P(x_N) = \prod_{i=1}^{N} P(x_i). \tag{2.1}$$

The Laplace transform method of solving Markov models is outlined by Shooman [31]. Solving the RAID 0 Markov model utilizing this method for components with identical constant failure rates yields the reliability equation below.

$$R(t) = \prod_{i=1}^{N} e^{-\lambda_i t} = exp(-\sum_{i=1}^{N} -\lambda_i t) = e^{-N\lambda t}. \tag{2.2}$$

In an example, if two identical hard drives with a failure rate of 1/100 failures per year are put in a RAID 0 configuration, the failure rate becomes 2/100 failures per year. Thus, to calculate the failure rate for RAID 0 excluding cloud drives, we can merely add the failure rates of each drive in the array.

2.6.3 RAID 1

RAID 1 requires a simple parallel reliability model, which is depicted in Figure 2.15. The figure represents a Markov model of a RAID 1 system with two drives, one mirroring the other.

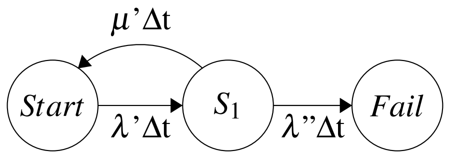
$$\begin{tikzcd}
Start \arrow[r, "\lambda'\Delta t"'] & S_1 \arrow[l, bend right, "\mu'\Delta t"'] \arrow[r, "\lambda''\Delta t"'] & Fail
\end{tikzcd}$$

Figure 2.16: Markov reliability models of traditional RAID 5 and local+cloud-RAID 4. S$_1$ represents degraded operation. This model assumes constant failure rates.

The reliability for this system follows the equation outlined by Shooman [31]:

$$R(t) = P(x_1 + x_2 + ... + x_N) = 1 - P(\bar{x_1}\bar{x_2}...\bar{x_N}). \tag{2.3}$$

For constant failure rate components, the reliability equation becomes:

$$R(t) = 1 - \prod_{i=1}^{N}(1 - e^{-\lambda_i t}). \tag{2.4}$$

2.6.4 RAID 2, RAID 3, and RAID 4

RAID 2, RAID 3, and RAID 4, excluding cloud drives, do not benefit from the architecture proposed in this thesis. Therefore, an analysis of these configurations from a mathematical perspective will not be performed.

2.6.5 RAID 5

RAID 5 with three disks can be modeled utilizing a serial reliability model, as illustrated in Figure 2.16. With RAID 5, λ' in this figure becomes 3λ, and λ" becomes 2λ. This is because three drives can fail at any time, and when one does, two drives are left to fail after the first drive fails.

$$P_{Start}(s) = \frac{s + 2\lambda + \mu'}{[s^2 + (5\lambda + \mu')s + 6\lambda^2]}. \tag{2.5}$$

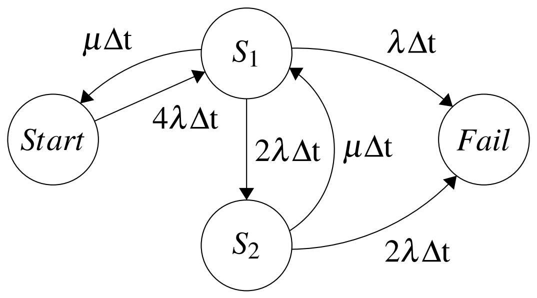

Figure 2.17: Traditional RAID 10 reliability model. S_1 and S_2 represent degraded operation states. This model assumes constant failure rates and one available repairman.

$$P_{S_1}(s) = \frac{3\lambda}{[s^2 + (5\lambda + \mu')s + 6\lambda^2]}. \tag{2.6}$$

$$P_{Fail}(s) = \frac{6\lambda^2}{s[s^2 + (5\lambda + \mu')s + 6\lambda^2]}. \tag{2.7}$$

The mean time to failure (MTTF) of this system is the limit of the sum of P(Start) and P(S_1) as s approaches 0, which results in:

$$MTTF = \frac{5\lambda + \mu'}{6\lambda^2}. \tag{2.8}$$

2.6.6 RAID 10

The reliability of RAID 10 follows the Markov model in Figure 2.17. Solving, the reliability equation for this model is:

$$P(s) = \frac{22\lambda^2 + 9\lambda s + 7\lambda\mu + s^2 + 2\mu s + \mu^2}{24\lambda^3 + (26s + 4\mu)\lambda^2 + s(9s + 7\mu)\lambda + s(s + \mu)^2}. \tag{2.9}$$

To obtain the MTTF of this configuration, we take the limit of this equation as *s* approaches 0, yielding:

$$MTTF = \frac{22\lambda^2 + 7\lambda\mu + \mu^2}{24\lambda^3 + 4\mu\lambda^2}. \tag{2.10}$$

The reliability analysis presented above is continued in the next chapter, where the focus shifts to local+cloud-RAID. The results of the analysis are then compared and conclusions are drawn about the relative reliabilities of local RAID and local+cloud-RAID.

CHAPTER 3. CLOUD-RAID AND A PROPOSED ARCHITECTURE: LOCAL+CLOUD RAID 4 / RAID-DP

In this chapter, a new architecture for local+cloud-RAID 4 is proposed. A previous paper outlines the fundamental innovation upon which this architecture is based [63]. The paper discusses how cloud and network-based storage can be utilized in different configurations in order to bolster storage parameters including reliability, availability, performance, and security.

3.1 Cloud-RAID

Cloud-RAID uses only cloud-based storage to construct a RAID group. For example, Cloud-RAID 0 has been discussed, implemented, and evaluated as a means to improve cloud storage performance and security [23]. Cloud-RAID 1 is a natural, common use of cloud-RAID, which consists of a mirroring of one cloud drive to another.

Local+cloud-RAID combines cloud storage and local storage in a single array. This structure has not received nearly the same scrutiny as cloud-RAID 0 or cloud-RAID 1. The base case for local+cloud-RAID 1 is the mirroring of a local disk to a cloud drive, which is an everyday practice. However, RAID 4, 5, 6, 10, and other levels have received little attention from researchers. In a previous paper [63], configurations with significant meaning for local+cloud-RAID are discussed, and the conclusion is made that RAID 4 presents a viable configuration on which to build an ideal local+cloud-RAID system. Such a system improves upon reliability, availability, security, performance, and storage efficiency as compared to other RAID levels.

3.2 Benefits of Cloud-RAID

Combining cloud storage and local storage provides many benefits not available to RAID systems consisting solely of local disks. These benefits include security, storage diversity, accessibility, maintenance, and data availability. While local storage typically offers better performance due to data locality, cloud storage provides benefits local storage cannot.

3.2.1 Security

Security is a benefit offered by cloud storage that is hard to match with local storage. However, data security has many facets, including protecting against:

- time loss and inconvenience,
- loss of privacy,
- loss of confidentiality,
- physical damage,
- financial implications and loss,
- reputational damage,
- legal implications, and
- the loss of trade secrets or other proprietary information.

Cloud storage can provide protection against all of these situations through the use of automatic encryption and data protection mechanisms, storage diversification, and redundancy [64].

3.2.2 Encryption

Encryption is a primary means by which cloud storage providers protect user data. In the event of theft or unauthorized access of data center hard drives, attackers must then decrypt data in order to recover user information. The goal of encryption within data centers is to make it computationally very difficult for an attacker to recover user data. Thus, encryption aids in assuring privacy and confidentially, and protecting against financial, reputational, or other damage. However, a key point is that merely encrypting data does not make it *impossible* for attackers to gain access to data, it only makes it more difficult. Given sufficient resources, such as those often available to governments, many means of encryption can be overcome with sufficient computational resources. Therefore, encryption by itself is not sufficient to protect privacy. Cloud-RAID can solve this problem by striping data across multiple cloud storage providers, ensuring that the unauthorized access of any one provider is insufficient to reconstruct the original data. By including the cloud in a RAID system, data can be protected and kept private, especially if the only data stored in the cloud is parity. This will be discussed in more detail in a later section.

3.2.3 Storage Diversity

Another benefit that cloud storage provides is storage diversity. This is a feature that is automatically implemented by many cloud storage providers. When storing data with vendors that implement this feature, vendors store independent copies of user data in multiple, geographically distinct locations. This helps to safeguard data in the event of emergency, natural disaster, and breaches at data centers, among other events. Diversifying storage is a means of not only providing redundancy, but ensuring greater availability of user data.

3.2.4 Availability and Reliability

Availability is the primary metric of interest when discussing cloud storage. Subscribers to cloud services are guaranteed minimum uptimes dependent on the provider. Microsoft ensures Azure has a 99.9% availability, which amounts to less than 43 minutes of downtime per month. Amazon guarantees an EC2 availability of 99.95%, which is less than 23.5 minutes of downtime per month. Both providers implement geo-replication of data, greatly mitigating the risk of any permanent loss of data [65–69].

Availability and reliability are important reasons that people move to cloud-based storage. Cloud storage is much more reliable than local data stored on HDDs, SSDs, or other flash drives. Anyone who has experienced data loss knows the frustration it causes, and preventing it is an important reason for cloud-based storage. By extending local RAID to include the cloud, data availability and reliability can be improved [70].

Because only availability numbers, rather than reliability numbers, are published by cloud storage providers, and because cloud storage providers employ reliability-enhancing techniques with any data stored, it is difficult to precisely quantify the reliability of cloud storage, or in other words, the probability that data becomes lost forever. Such events do, however, occur. For example, in 2011, Amazon's compute services suffered a catastrophic event caused by a poorly deployed update and inherent bugs within EC2. These bugs were made visible by means of a botched update that caused extensive downtime [71]. Additionally, this event caused permanent data loss within Amazon's online computing services for customers hosting data within Amazon's cloud [72]. Furthermore, the closure of cloud service providers due to bankruptcy or the loss of data due to malicious intent are both known possibilities [7,17,19,23]. Permanent data loss caused by catastrophic events can be prevented by utilizing erasure coding in the cloud. Erasure coding consists of breaking up data into chunks, expanding each chunk with redundant data, and storing those chunks in different locations [23]. With the intrinsic reliability of cloud storage, the probability of permanently losing an arbitrary piece of data within the cloud is extremely unlikely. The author is unaware of any study quantifying the current risk of permanent data loss among cloud providers, or of any estimates of the total amount of data historically hosted in reliable cloud storage versus the amount that has been hitherto permanently lost through various means, nefarious or otherwise.

Given the immense amount of data stored in the cloud, and the relative infrequency of catastrophic data loss events, the probability of data loss per amount of data stored is immeasurably small, although unquantified.

3.2.5 Accessibility

Online data storage provides a level of accessibility to users that is hard to match with local storage. Advanced users can easily make their home data available to them while not at home, but most people rely on devices such as a USB flash drive or portable hard drive. These devices can have dubious reliability, but they nonetheless aid in accessibility. Online storage provides the benefit of reliable, accessible storage without the need to carry around a device meant exclusively for storing data.

3.2.6 Maintenance

The lack of maintenance required by users of online storage makes it another attractive option for those seeking an easy, usable storage solution. Over time, hard disks and other storage media can become unreliable. Data centers constantly maintain user data by providing geographical diversification, storage redundancy, and monitoring of storage arrays. This is all work that does not have to be done by individual users, and it happens behind the scenes at storage centers.

3.3 Local+Cloud-RAID

All of the benefits provided by cloud storage can be combined with the performance benefits of local storage through a wise use of RAID and an appropriate architecture. Local+cloud-RAID is already widely implemented as RAID 1, a cloud-based backup of a local drive. It is widely used to mitigate data loss by means of redundancy in the cloud. Additionally, no severe performance penalties are incurred for maintaining a backup, as long as the backup is not strictly required to be kept in real-time. This configuration provides greater data availability than that of the cloud medium used, or an almost 100% availability of data. The only situation where data unavailability occurs is when the local drive fails and the cloud backup is also unavailable. This is the only currently common use of local+cloud-RAID.

In this section, I outline two local+cloud-RAID architectures that can bolster security, privacy, confidentiality, and performance: local+cloud-RAID 4 and local+cloud-RAID-DP.

3.3.1 Local+Cloud-RAID 4

Local+cloud-RAID 4 combines data striped across local storage and a parity drive in the cloud. Suppose a RAID 4 array is created with two local drives and one cloud drive. This configuration is depicted in Figure 3.1. The cloud drive is selected to serve as the dedicated parity drive for the array. Because dedicated parity drives incur more write cycles than the other disks in a RAID 4 array, they can degrade faster due to the wear caused by writing to a disk. By migrating the parity disk to the cloud, the wearing effected caused by repeated writes is mitigated. Users need not worry about causing wear to their own devices when cloud storage is used instead of their local disks. Traditional RAID 5 and RAID 6 incur many extra write cycles per drive per unit of data written due to interspersed parity. However, this new proposed configuration of RAID 4 mitigates this problem because all extra parity writes are performed to the cloud. The writes do not cause wear to local, physical drives, nor do they degrade read performance of the system.

Additionally, complete and quick reads can be performed exclusively with the local drives, as the parity drive only is read when the array runs in degraded mode and is being reconstructed. Thus, RAID 0 read performance levels can be achieved in this configuration. Parity information is not interspersed with the data, as it is with RAID 5 and RAID 6, so precious storage space is conserved among local drives. Components are often sensitive to excessive write cycles, which cause wear. This is why modern storage devices often utilize error correcting codes and software to help drives evenly distribute wear [73–75].

One potential challenge in this system is the difficulty in keeping up with writes to the array. Because parity information is generated on the fly in real-time, this extra information would need to be written at an equal pace to the cloud as data is written to local drives. We would not want to outstrip the ability of the system to keep up with repeated writes.

This drawback is not insurmountable. In fact, the RAID implementation from Lime Technology called unRAID combines an embedded system and a fast write cache in order to keep up with writes to a dedicated parity drive [60]. In Chapter 5 an architecture extending this configuration to include dedicated hardware to handle local+cloud-RAID 4 is presented.

The ideal configuration for local+cloud-RAID 4 consists of an array of local storage configured as RAID 0 with a wrapper that handles parity calculation across the array, parity updating, and array management. The wrapper – implemented either with software or specialized hardware – handles synchronizing parity information with cloud storage provider(s) and reconstruction of the local RAID 0 array in the case of local disk failure.

This configuration, in addition to providing improved reliability and performance, provides an additional benefit of security as compared to more traditional RAID configurations. When only parity information is stored in the cloud, not only is it encrypted, but even if data is accessed without authorization, and an attacker manages to somehow decrypt the data, the recovered data, by itself, would only contain random, useless bits. This ensures perfect privacy and confidentiality, regardless of encryption. In fact, encryption becomes unnecessary when only parity is stored in the cloud.

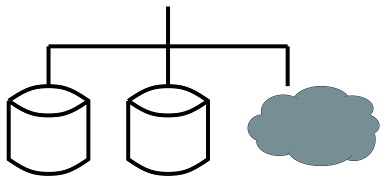

Figure 3.1: Local+cloud-RAID 4 with dedicated cloud parity. The local disks are configured in RAID 0. Parity is calculated and stored in the cloud.

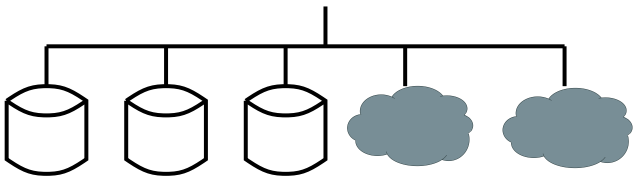

Figure 3.2: Example implementation of proposed local+cloud-RAID-DP with two dedicated cloud parity drives. Two local drives can fail and data is still recoverable.

3.3.2 Local+Cloud-RAID-DP (Double Parity)

The same benefits that the local+cloud-RAID 4 architecture provides can be extended to include multiple levels of parity. Figure 3.2 shows an example of this array. This configuration provides more benefits than having a single layer of parity. For each layer of parity calculated for an array, the array can tolerate that many simultaneous disk failures before repair.

Consider a local+cloud-RAID-DP system consisting of two local drives configured as RAID 0, with two independent layers of parity stored to two separate cloud storage providers. If attackers manage to compromise a single storage provider, they cannot recover the original data due to it only containing parity information. Even if attackers manage to identify and compromise both of the cloud storage parity locations of the array, they still must break the unique encryption of both of the cloud storage providers in order to recover the original data. This configuration also survives the loss of *all* of the local drives, because it enables the recovery of the entire original array, with no loss to integrity or risk to privacy or confidentiality. This configuration results in an extremely reliable, high-performing, and secure RAID system. Even the most high-performing RAID systems on the market are not backed by secure, private cloud storage system. Additionally, none of them can tolerate the loss of *all* storage devices within the array. This is a vast improvement over existing systems in terms of reliability and security.

3.4 Cloud-RAID and Local+Cloud-RAID Reliability Analysis

This section extends upon the analysis conducted in Chapter 2 by adding cloud disks to various RAID configurations. These analyses are presented in a previous paper [63]. It is assumed that the reliability of storing data in the cloud is orders of magnitude greater than the reliability of storing information only locally. Thus, when cloud drives are paired with local drives for RAID reliability analysis, the component that cloud drives contribute to adversely affect reliability are negligible and therefore ignored. For mathematical analysis, cloud storage is assumed to be infinitely more reliable than local drives when they are paired together, due to the unquantifiably minute contribution the cloud contributes to permanent data loss as compared to the contribution of local drives.

3.4.1 Cloud-RAID 0

Cloud-RAID Level 0 has been described previously in various implementations. Performance and security benefits have been documented as well as the expected and actual performance improvements of implementing such a system. This configuration is feasible as a method to increase the performance of cloud storage in terms of performance, availability, reliability, and security [15, 16, 18, 19, 23–26]. Availability, a measurement of uptime, is relevant for cloud-based storage. To calculate the availability of this system, the availabilities of the utilized cloud storage media are merely multiplied.

3.4.2 Local+Cloud-RAID 0

This RAID configuration is infeasible. Read speeds and writes speeds are bottlenecked by the cloud drive within the array, as the read and write performance differential between local and cloud storage is large.

3.4.3 Cloud-RAID 1

This configuration consists of a cloud-based backup of another cloud drive. It is sufficient to say that it has much improved data availability relative to storing data at a single cloud location, which is already guaranteed by multiple providers to be 99.9% [65–69]. A model for this is equivalent to a model for a RAID 1 array that contains only local drives, with availability numbers used in calculations rather than individual drive failure rates.

3.4.4 Local+Cloud-RAID 1

This RAID configuration consists of a cloud-based backup of a local drive. It is widely used to mitigate data loss by means of redundancy in the cloud. Additionally, no severe performance penalties are incurred for maintaining a backup, as long as the backup is not strictly required to be kept in real-time. This configuration provides greater data availability than that of the cloud medium used. The only situation where data unavailability occurs is when the local drive fails and the cloud backup is also unavailable. This configuration does not merit a model due to its simplicity.

3.4.5 Local+Cloud-RAID 4

Reliability for this system can be calculated with the aid of a Markov model. See Figure 2.16. For local+cloud-RAID 4, λ' = 2λ and λ" = λ. This model was solved by Shooman [31]. The probability, in the Laplace domain, of being in each state is:

$$P_{Start}(s) = \frac{s+\lambda+\mu'}{[s^2+(3\lambda+\mu')s+2\lambda^2]}. \tag{3.1}$$

$$P_{S_1}(s) = \frac{2\lambda}{[s^2+(3\lambda+\mu')s+2\lambda^2]}. \tag{3.2}$$

$$P_{Fail}(s) = \frac{2\lambda^2}{s[s^2+(3\lambda+\mu')s+2\lambda^2]}. \tag{3.3}$$

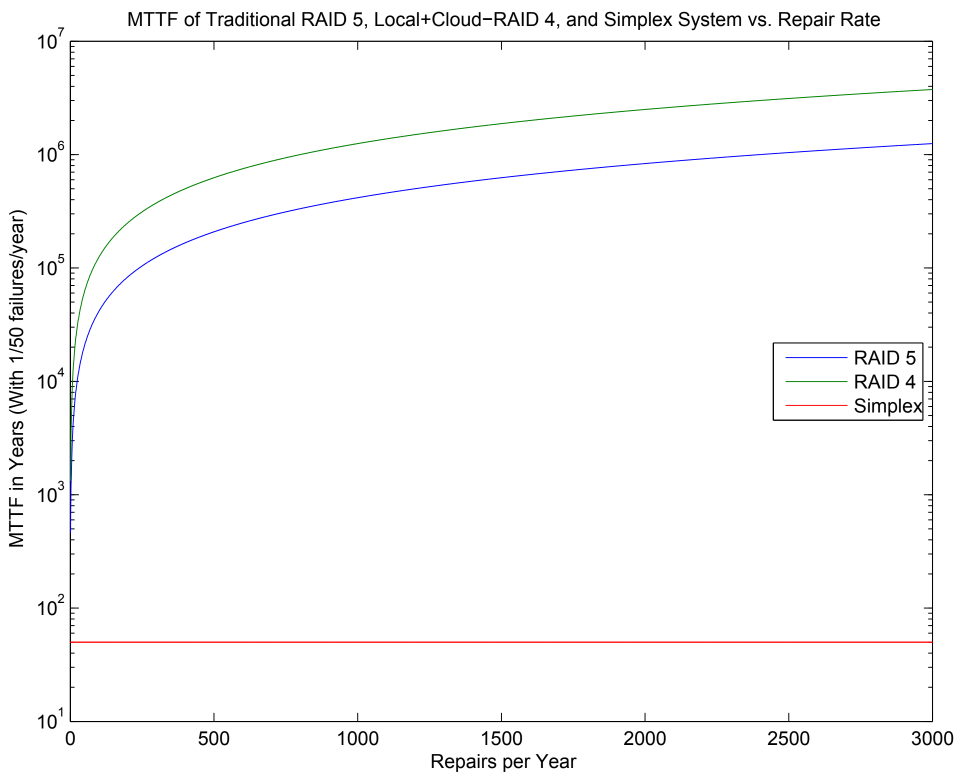

Figure 3.3: MTTF of RAID 5 and cloud-RAID 4 versus Repair Rate

The MTTF of this system is the limit of the sum of P(Start) and P(S_1) as s approaches 0, which results in:

$$MTTF = \lim_{s\to 0}(P_{Start}(s) + P_{S_1}(s)) = \tag{3.4}$$

$$\lim_{s\to 0}\frac{s+3\lambda+\mu'}{[s^2+(3\lambda+\mu')s+2\lambda^2]} = \frac{3\lambda+\mu'}{2\lambda^2}. \tag{3.5}$$

Comparing to equation 2.8, it is evident that local+cloud-RAID 4 provides superior reliability than RAID 5 with no cloud disks. One potential alternative to this system is creating a backup (RAID 1) or calculating parity to store in the cloud while only maintaining a RAID 0 array locally. This enables very high local performance while maintaining reliability of data.

Figure 2.16 and the above discussion lead to an interesting result: local+cloud-RAID 4 reliability is always higher than traditional RAID 5 reliability for equivalent constant failure rates

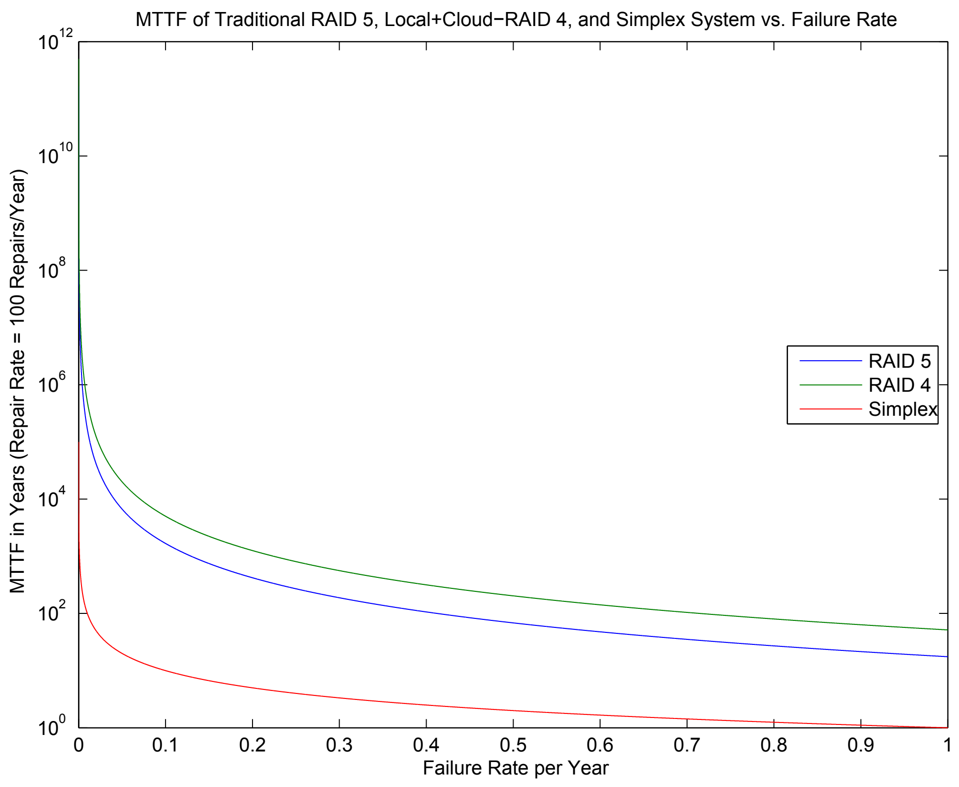

Figure 3.4: MTTF of RAID 5 and cloud-RAID 4 versus Failure Rate

between drives. The reliability improvement is $(3\lambda / 2) / (5\lambda / 2)$. Assuming a realistic failure rate of $\lambda = 0.02$ failures / year, and $\mu = 100$ repairs / year, the reliability improvement is approximately a factor of 3. Figure 3.3 plots the MTTFs of local+cloud-RAID 4, traditional RAID 5, and a simplex system against increasing repair rates, holding the failure rate constant at 1 failure per 50 years. Figure 3.4 plots these MTTFs against failure rates, holding repair rate constant at 100 repairs/year.

3.4.6 Local+Cloud-RAID 10 / 01

Consider a stripe of mirrors array (RAID 10, Figure 3.7) where mirrors exist in the cloud. Compare this to a mirror of stripes (RAID 01, Figure 3.8) where the mirrors also exist in the cloud. These configurations, interestingly, are practically identical functionally as well as from a reliability perspective when two cloud drives and two local drives are used. The reliability model for these configurations is the same as in Figure 2.16, with $\lambda' = 2\lambda$ and $\lambda'' = \lambda$.

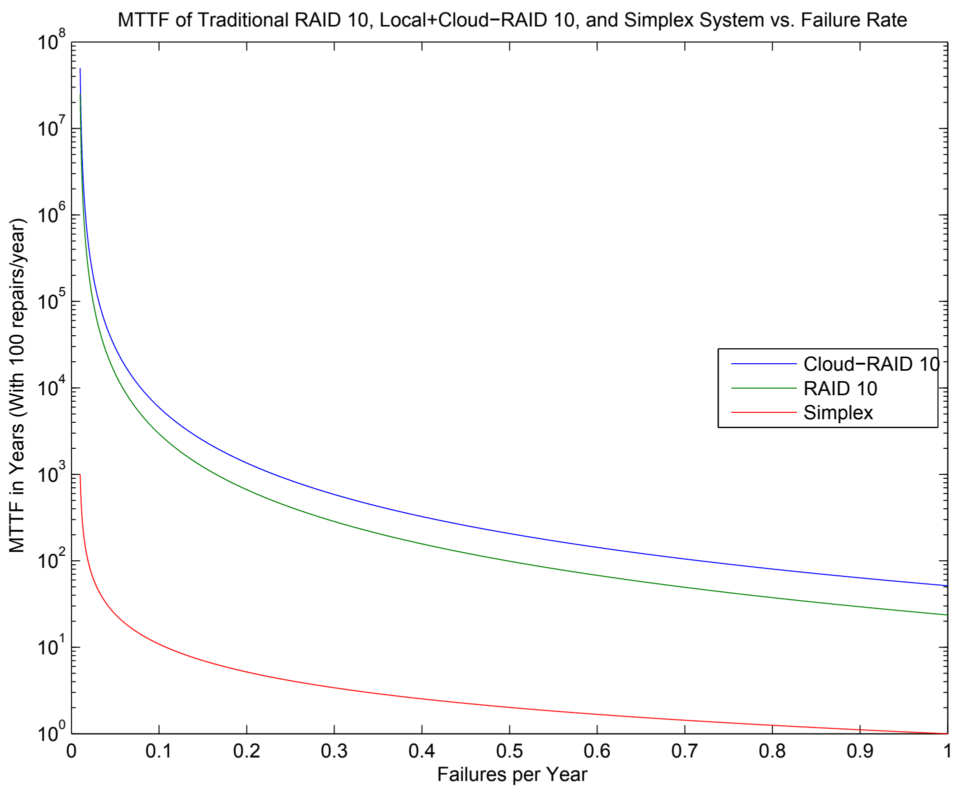

Figure 3.5: MTTF of RAID 10 and cloud-RAID 10 versus Failure Rate

Comparing the MTTF in equation 2.10 to the MTTF in equation 3.4, an interesting result is seen. Plotting the MTTF of RAID 10 and cloud-RAID 10 against varying drive failure rate while holding repair rate constant, Figure 3.5 shows that cloud-RAID 10 always has the highest MTTF. Similarly, this is true if failure rates are held constant and repair rates are varied, as illustrated in Figure 3.6. This configuration offers RAID 0-like performance.

As Figure 3.5 shows, the MTTF for cloud-RAID 10 is always higher than for traditional RAID 10. The reliability improvement is obtained by dividing equation 3.5 by equation 2.10. Assuming a reasonable failure rate of 0.02 failures / year and a repair rate of 100 possible repairs / year, the reliability improvement is approximately 2.

3.5 Summary of Local+Cloud-RAID Benefits

Table 3.1 shows a comparison of various levels of traditional RAID, online mirroring, and local+cloud-RAID. Each level possesses one or more desirable characteristics. These characteristics

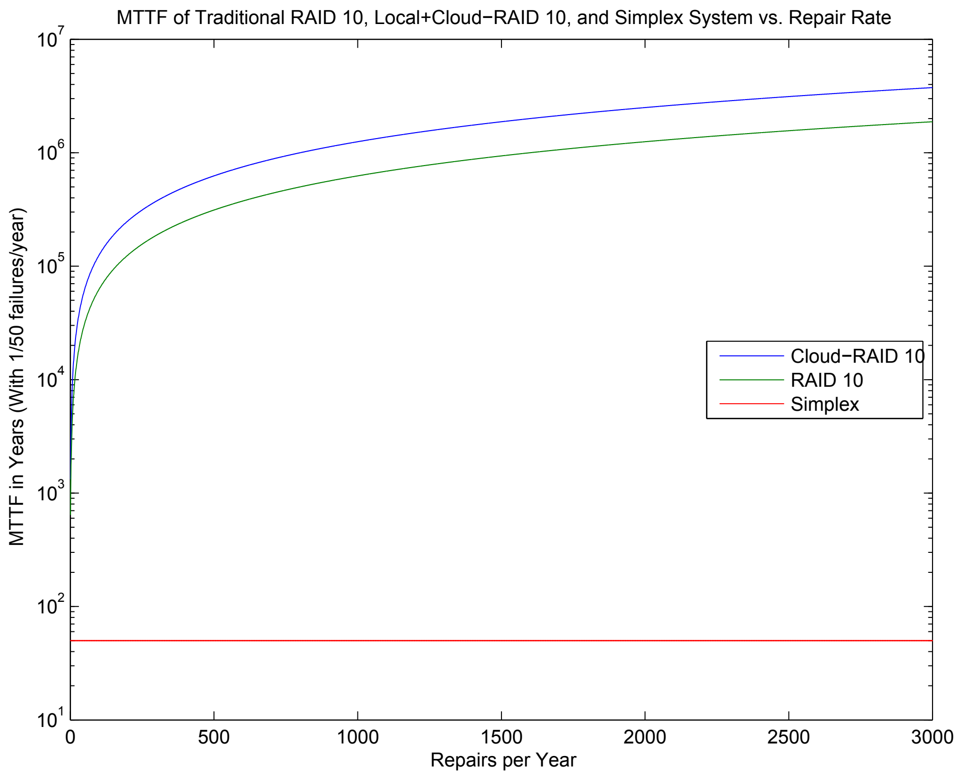

Figure 3.6: MTTF of RAID 10 and cloud-RAID 10 versus Repair Rate

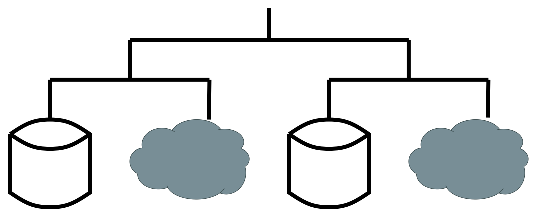

Figure 3.7: Local+cloud-RAID 10 (stripe of mirrors) with cloud mirroring

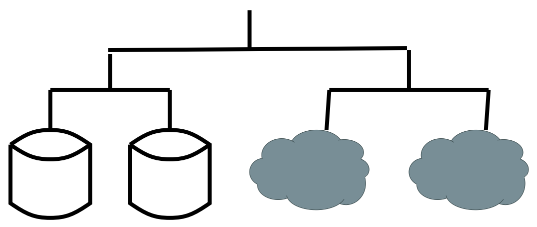

Figure 3.8: Local+cloud-RAID 01 (mirror of stripes) with cloud mirroring

Table 3.1: A Relative Comparison of Various RAID and Local+Cloud RAID Levels with D Data Disks [1]

	RAID 0	**RAID 1**	**RAID 4**	**RAID 5**	**RAID 6**	**RAID 10**	**Cloud Mirror**	**Local+Cloud-RAID 4**	**Local+Cloud-RAID-DP**
MTTF	Lowest	Very High	High	High	Very High	Very High	Highest	Very High	Highest
Total Number of Disks	D	2D	D+1	D+1	D+2	2D	N/A	D	D
Usable Storage Capacity	100%	50%	((D-1)/D)%	((D-1)/D)%	((D-2)/D)%	50%	100%	100%	100%
Potential for Privacy Loss	Low	Low	Low	Low	Low	Low	High	Low	Low
Read Performance	Highest	Very High	High	High	High	Very High	Lowest	High	High
Write Performance	Highest	Very High	Low	High	High	Very High	Lowest	High	High

are MTTF, total number of disks, usable storage capacity, privacy, read performance, and write performance. Local+cloud-RAID 4 and local+cloud-RAID-DP possess many of these desirable attributes compared with other configurations. This section discusses the benefits of the new architectures compared with existing RAID.

3.5.1 MTTF

Patterson et al. [1] analyze the MTTF for various traditional RAID levels. When combined with cloud-based redundancy, the MTTF of an array further increases, as previously discussed. This means that local+cloud-RAID 4 and local+cloud-RAID-DP have the highest MTTFs of all analyzed arrays due to the rarity of permanent data loss in cloud storage among large data providers.

3.5.2 Total Number of Disks and Usable Storage Capacity

These two attributes are interconnected; in general, where more physical disks are required, usable storage capacity decreases. This means that when mirroring is used, in the case of RAID 1, the array contains less overall usable storage compared to RAID 4 or RAID 5, which use parity instead of mirroring. Local+cloud-RAID 4 and local+cloud-RAID-DP can use the entirity of local physical disks for data, because all redundancy is stored in the cloud.

3.5.3 Privacy

Privacy is a desirable characteristic for any type of storage, whether local or online. Privacy entails revealing any personally identifiable and potentially damaging information about oneself. When data is stored only locally, privacy can much more easily be guaranteed because an attacker must have physical access to the storage device in order to retrieve any information. However, when data is stored online, this is not the case. If an attacker gains access to an account by unauthorized means, information can be acquired without physical disk access. Thus, privacy concerns are much more relevant for online storage than for local storage.

When a disk is mirrored to a cloud-based storage location, in full or in part, full file information is typically stored in its entirety on at a single provider, often unencrypted and only protected by a password. If an attacker gains access, they have full access to all information stored in each file. Privacy concerns are problematic in the case of cloud-based mirroring of local storage. Local+cloud-RAID 4 and local+cloud-RAID-DP solves this problem. This is accomplished through not storing complete file data at any individual cloud storage location. Local+cloud-RAID-DP stores each layer of parity in its own dedicated cloud storage location. Thus, an attacker needs to know which cloud storage providers are in use for any particular array, and also the login information for each layer of parity. If an attacker manages to compromise one layer of parity, there is not enough information revealed to the attacker to reconstruct the original data, because it is only parity that is calculated from many disks. In the case of local+cloud-RAID 4, there is not sufficient information stored online in order to reconstruct the original data, so an attacker must also have access to the physical data disks, preserving privacy.

3.5.4 Read Performance

Read performance of local+cloud-RAID 4 and local+cloud-RAID-DP in a software implementation is similar to traditional RAID 4 and RAID 5, as shown in the next chapter. Parity disks are not needed in order to service read requests. In the hardware implementation discussed in Chapter 5, read performance can be further increased to the level of RAID 0, which offers the highest theoretical performance.

3.5.5 Write Performance

Because local+cloud-RAID 4 and local+cloud-RAID-DP utilizes write caching to update cloud-based parity information, overall write performance can be much improved. The array must still calculate parity, as is the case for traditional RAID 4 and RAID 5, but the local+cloud-RAID arrays can perform faster in hardware implementations due to fewer bytes being stored locally and the relative speed of the write cache compared to local disk write speed. The write cache allows parity, once calculated to be put in a “fire and forget” FIFO that allows the array to continue with operations while parity is being written to the cloud.

3.6 Costs of Local+Cloud-RAID

This architecture requires a user to have an Internet connection and a subscription to one or more cloud storage providers. The process of streaming parity information on a consistent basis to the cloud consumes upload bandwidth, and much download bandwidth is consumed in the event an array needs to be reconstructed.

CHAPTER 4. METHODOLOGY, PROOF OF CONCEPT, AND PERFORMANCE RESULTS

Local+cloud-RAID 4 can be implemented with either software or hardware. The simplest and easiest implementations are in software. Many RAID solutions are implemented in software, whereas hardware implementations are much more expensive, requiring extra physical components inside a system. In this chapter a software implementation of local+cloud-RAID 4 using Linux and freely available tools is presented. In the next chapter, a more complex hardware architecture is presented.

4.1 Methodology

In this section, the methodology to replicate the software local+cloud-RAID 4 solution is presented. This section describes the equipment used, the software tools to construct local+cloud-RAID 4, the step-by-step instructions to replicate the setup, and the method by which results were measured.

4.1.1 Equipment

Various RAID configurations were implemented using an HP dv6-7029wm laptop with an AMD A8-4500M processor and 8GB of ram. For storage devices, three identical ADATA 8GB UV128 USB 3.0 flash drives were utilized. A Targus ACH122USZ USB 3.0 hub was used which has 3 USB 3.0 slots into which the USB flash drives were inserted. The ACH122USZ was connected to one of the USB 3.0 slots on the dv6-7029wm.

4.1.2 Software

The OS used was Ubuntu 16.04.1 LTS. The mdadm tool built into Linux was used to configure various RAID arrays. Additionally, the gnome-disks utility was utilized for array benchmarking. The tmpfs tool was used to create a ramdisk which was then mounted.

4.1.3 Ramdisks

Ramdisks are a feature built into Linux that allow a portion of RAM to function as a mounted disk drive. They can either be created with tmpfs from the console or at boot time by editing /etc/fstab to include tmpfs. Furthermore, Ubuntu by default allocates ramdisks at /dev/ram, and the parameters of these ramdisks can be configured inside the .config file at /boot/config-* by editing the lines CONFIG_BLK_DEV_RAM, CONFIG_BLK_DEV_RAM_COUNT, and CONFIG_BLK_DEV_RAM_SIZE.

As long as the system is powered on, the ramdisk can function as a very fast alternative to traditional storage. It can even be included in a RAID array as a device with mdadm. The only downfall is that RAM is volatile, and any contents of the ramdisk that remain after powering off the system are erased.

4.1.4 Setup

Here are step-by-step instructions on how to replicate the setup for each RAID array tested.

1. Edit /etc/fstab to remove the OS swap space to disable paging for more accurate benchmarking.
2. Configure a ramdisk by editing /boot/config to create 1 ramdisk (/dev/ram0) with a size of at least 2GB.
3. Install lsyncd (live syncing daemon). This enables monitoring a directory structure for changes and syncing them close to real time to another location.
4. Reboot the system.
5. After reboot, swap should be disabled and the size of /dev/ram0 should match /boot/config.

6. Set up a mount of cloud storage sufficient for the size of the desired RAID configuration. This will be used by lsyncd to synchronize the dedicated parity ramdisk to the mounted cloud storage drive.
7. Format the ADATA USB drives as an ext4 filesystem (standard Linux filesystem).
8. Format /dev/ram0 as an ext4 filesystem.
9. Create /dev/md/myRAID.
10. Create a RAID array for benchmarking by executing: sudo mdadm –create –verbose /dev/md/myRAID –level=X –raid-devices=X USB1_location USB2_location
11. Mount /dev/ram0.
12. (For RAID-4 only) Use lsyncd to sync the dedicated parity disk to the cloud storage mount by running lsyncd -rsync /path-to-ramdisk-mount /path-to-cloud-storage-mount.

The above steps were taken for seven distinct configurations, outlined below, and data on read and write performance was gathered using the gnome-disks benchmarking utility. The selected configurations provide useful data against which to compare local+cloud-RAID 4.

- One ramdisk; mount /dev/ram0 and run the gnome-disks benchmarking utility on the ramdisk alone.
- One ADATA USB drive; mount /dev/sdb (or wherever the ADATA USB drive is located) and run the gnome-disks benchmarking utility.
- RAID 0 with two ADATA USB drives; sudo mdadm –create –verbose /dev/md/myRAID –level=0 –raid-devices=2 /dev/sdb /dev/sdc.
- RAID 4 with three ADATA USB drives; sudo mdadm –create –verbose /dev/md/myRAID –level=4 –raid-devices=3 /dev/sdb /dev/sdc /dev/sdd. Note that according to the system manual entry for md, the last device in the –raid-devices option list is the dedicated parity drive.

Table 4.1: gnome-disks Benchmark Data

Array Benchmarks	Average Read (MB/s)	Average Write (MB/s)
Ramdisk Only	1700	605
Single USB Drive	36.2	4.4
RAID 0 w/ 2 USB	68.1	7.1
RAID 4 w/ 3 USB	62.0	4.8
RAID 5 w/ 3 USB	69.2	6.8
RAID 4 w/ 2 USB + ramdisk	68.0	6.7
RAID 5 w/ 2 USB + ramdisk	65.7	6.7

- RAID 5 with three ADATA USB drives; sudo mdadm –create –verbose /dev/md/myRAID –level=5 –raid-devices=3 /dev/sdb /dev/sdc /dev/sdd.
- RAID 4 with two ADATA USB drives and a ramdisk as dedicated parity drive; sudo mdadm –create –verbose /dev/md/myRAID –level=4 –raid-devices=3 /dev/sdb /dev/sdc /dev/ram0.
- RAID 5 with two ADATA USB drives and a ramdisk; sudo mdadm –create –verbose /dev/md/myRAID –level=5 –raid-devices=3 /dev/sdb /dev/sdc /dev/ram0.

4.2 Results

Using the gnome-disks disk benchmarking tool, the average read and write performance for each configuration was measured over 100 samples of 50 MB/sample. This sample size smooths out noisy data, getting more accurate steady state measurements of throughput. Due to the transfer speeds involved, 100 was chosen as the number of samples for each benchmark in order to balance the need to accurately measure average throughput and the time taken for each test. The results are contained in Table 4.1. When benchmarking data multiple times, exact measurements may change due to various variables present within a running system, but general trends remain consistent.

4.3 Discussion

The results of this experiment are as expected. The ramdisk has orders of magnitude higher performance than a single USB drive. The single USB drive performed worst, and the highest performing arrays were RAID 0, RAID 4 with 2 USB drives and a ramdisk parity drive, and RAID 5. Traditional RAID 4 performs worse for reading and writing than traditional RAID 5, but a key difference is when the ramdisk is added as parity in RAID 4. This enables RAID 4 to perform similarly to RAID 5, because the dedicated parity disk, which is normally a performance bottleneck for writes, performs much faster than the other disks in the array. This is because the dedicated parity disk in this configuration is RAM.

The lsyncd utility allows for easy replication of the dedicated RAID 4 ramdisk to the cloud. Thus, through the combination of mdadm, ramdisks, and lsyncd, we can create a lightweight and functional local+cloud-RAID 4 in software. Adding lsyncd to synchronize the dedicated parity ramdisk to the cloud does not inhibit the throughput of the local disks in the system. This is because the ramdisk is a different storage device than the rest. RAM has much higher throughput, and when data is streamed to the cloud from RAM, the system does not consume disk bandwidth.

4.3.1 Benefits of Implementation

The benefits of the presented software local+cloud-RAID 4 implementation are many. Firstly, all of the tools are freely available. Next, RAID 0-like performance can be achieved. RAID 4 is the same as RAID 0 with an added parity disk. This implementation matches RAID 0 with fast parity generation that can be kept in RAM and streamed to the cloud mount using lsyncd, increasing reliability greatly. Reliability is vastly improved with the redundancy added by parity. Lastly, all data stored in the cloud is guaranteed to remain private; there is no possible means for an attacker who gains access to my cloud storage account to recover the data on my USB drives, because only parity information is stored.

4.3.2 Drawbacks of Implementation

The primary drawback of this implementation is that it requires a large dedicated portion of RAM to be utilized as a buffer, which prevents its use by other programs. The use of a ramdisk is necessitated by the limitations of mdadm. The mdadm utility does not allow asynchronous generation and storage of redundant information; when data is stored to the parity disk, it is stored in real time. This strict adherence to synchronicity can hinder performance, but is the primary reason for using a ramdisk in this implementation. If timing requirements for parity generation were to be relaxed, parity can be generated and streamed in the background after computationally intensive data operations are finished, and the ramdisk can be eliminated. The number of writes can be reduced by waiting for the final parity values to be calculated after a series of writes, and then only storing this data to the cloud.

Additionally, mdadm does not support an array configuration with an arbitrary number of dedicated parity drives; the limit is one. The mdadm utility supports RAID 6, which offers two layers of parity for protection against dual disk failures, but these layers of parity are distributed rather than dedicated. Therefore, I see no way to create local+cloud-RAID-DP using mdadm.

CHAPTER 5. A HARDWARE-BASED LOCAL+CLOUD-RAID ARCHITECTURE

In order to improve performance and offload CPU processing, many RAID controllers are implemented in hardware. Such RAID controllers can be very expensive, ranging from approximately \$50 to more than \$70,000 for a Proavio DS240FS. Hardware RAID controllers offer performance benefits over software RAID. In this chapter, benefits and drawbacks to existing hardware RAID controllers are discussed, as well as features that, if implemented, can enable very high performing local+cloud-RAID systems. An architecture that incorporates the benefits of local+cloud-RAID 4 and hardware RAID is presented and discussed in this chapter.

5.1 Hardware RAID Controller Benefits

Hardware RAID has some benefits over software RAID. While generally offering higher read and write performance than equivalent software implementations, hardware RAID offers the following additional benefits [76]:

- Data protection during boot, since the array does not rely on software to operate.
- Array available at boot.
- Write-back caching.
- Better protection of data in the event of power loss.
- No OS required.
- Easy OS migration.
- Protection against viruses.

If it were possible to combine these benefits with the benefits of a local+cloud-RAID system, a system that combines the highest level of performance, reliability, and security surpassing that of any existing RAID system can be created. This is due to the recoverability and privacy offered by the cloud and the performance improvement offered by hardware, as discussed in Chapter 3. In section 5.5, an architecture that combines these benefits is proposed. This architecture is a major contribution of this thesis.

5.2 Hardware RAID Controller Drawbacks

Not everything about hardware RAID is an improvement over software RAID. Unless specifically designed for reconfigurability, hardware, by nature, is not configurable; hardware RAID controllers are generally designed for a fixed number of disks and for fixed RAID architectures. This is because they are designed to interact with devices within a local storage environment. All devices within arrays created by these controllers must be physically connected to the RAID controller, meaning the disks are not connected via a network.

Unlike software RAID, which currently has the capability to include networked and cloud storage locations, hardware RAID cannot. This renders current hardware RAID controllers impractical for use with local+cloud-RAID, because they have no physical access to the storage locations and no network connectivity or Internet access.

Lastly, the development of a hardware RAID controller is a multi-year effort. The development and measurement of such a system exceeds the scope of this thesis.

5.3 A New Class of Hardware RAID Controller

A RAID controller that not only handles the tasks that existing RAID controllers perform, such as array creation, monitoring, reconstruction, and management, but that also combines the ability to connect to networks and have Internet access, provides the ability to combine the benefits of hardware RAID with the benefits of local+cloud-RAID. This is possible through the use of dedicated digital hardware combined with an embedded microprocessor that communicates with cloud storage application program interfaces (APIs) via HTTP or JSON.

Table 5.1: Common cloud storage provider API support.

	XML	HTTP / REST	SOAP	JSON
Amazon S3	Yes	Yes	Yes	No
Amazon Cloud Drive	No	Yes	No	Yes
Box	No	Yes	No	Yes
Dropbox	No	Yes	No	Yes
Google Drive	No	Yes	No	Yes
OneDrive	No	Yes	No	No
SugarSync	Yes	Yes	No	No
MediaFire	Yes	Yes	No	Yes
CloudMine	No	Yes	No	Yes
Rackspace	Yes	Yes	No	Yes
4shared	Yes	via SOAP	Yes	No
DigitalBucket	Yes	Yes	No	No
MEGA	No	Yes	No	Yes
Filepicker.io	No	No	No	Yes
EVS iDrive	Yes	Yes	No	No
Apple iCloud	No	via CloudKit	No	No
CloudApp	No	Yes	No	Yes
Verizon Personal Cloud Storage	No	Yes	No	Yes
FilesAnywhere	Yes	via SOAP	Yes	No

5.4 Cloud Storage APIs

This section outlines the available APIs for many commonly used cloud storage providers. This list is not comprehensive, as numerous lesser-known providers exist, and as the cloud storage landscape is constantly under flux. See table 5.1 for a list of cloud storage providers and the programming interfaces they provide within their respective APIs. XML (Extensible Markup Language), HTTP (Hypertext Transfer Protocol) / REST (Representational State Transfer), SOAP (Simple Object Access Protocol), and JSON (JavaScript Object Notation) are ways APIs can communicate with software. REST is implemented over HTTP, and HTTP is universally supported by the largest and most well-known providers [77].

This is fortunate from the perspective of embedded systems. C is the most widely used programming language for embedded system design. Many C libraries exist for communication over HTTP, and a commonly used library is called libcurl [78]. This enables a hardware-based RAID controller that utilizes a microprocessor running C code to communicate over HTTP with cloud storage providers.

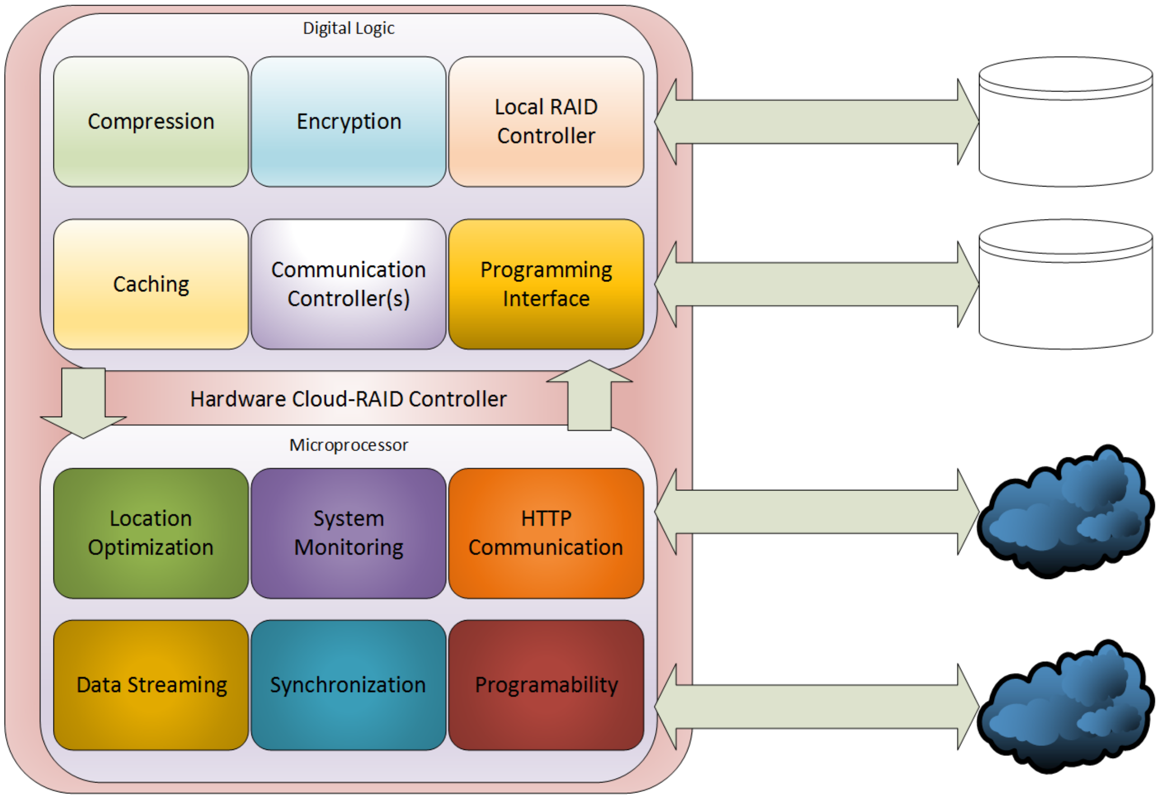

Figure 5.1: Hardware Cloud-RAID Controller Architecture

5.5 A New Networked RAID Controller Architecture

Figure 5.1 shows a graphical representation of the proposed hardware-based cloud-RAID controller which contains two local storage disks and two dedicated cloud parity drives. The system consists of the controller along with the attached local storage devices and the cloud storage located on the network. The controller itself consists of two parts: custom digital logic and an embedded microprocessor. Each of these subsections performs specific tasks to enable the system to work optimally.

5.5.1 Digital Logic

The digital logic contains functionality to support traditional RAID controller functionality as well as additional functionality needed to support the microprocessor and user programmability.

The Local RAID Controller module supports the creation, monitoring, repair, parity calculation, array reconstruction, and all supporting functions needed to create and maintain RAID 0, 1, 4, 5, 6, and 10 arrays, which reflect the levels supported by common existing hardware RAID controllers. Additionally, when local storage is configured for RAID 0 containing N disks, the user is presented with the ability to specify any number of dedicated parity disks less than or equal to N. These disks can be located locally, on a network, or in the cloud. In the case that the disks are located in the cloud, the user is able to provide login information for all selected cloud storage locations. Then, the microprocessor handles streaming of the calculated parity information to the selected cloud location(s). This case utilizes the full potential of the cloud-RAID controller. The local RAID controller also handles storage requests from the CPU through PCI-express, SATA, or other communication protocol.

The Communication Controller consists of one or more controllers implementing Ethernet, PCI-express, and/or SATA. PCI-express is the protocol for the highest performing storage devices on the market. With an Ethernet controller located on the common PCI-express bus, the cloud-RAID controller can communicate over the Internet without the need of an additional Ethernet controller on the device. This would eliminate the need for an Ethernet controller. SATA, however, is a master/slave interface. SATA does not have the ability to communicate via Ethernet under any circumstance. When implementing a SATA controller, an Ethernet controller must also be implemented on-chip. To summarize, the system needs to implement one of the following combinations of communication controllers:

1. A SATA controller and an Ethernet controller.
2. A PCI-express controller only, in the case of having an Ethernet controller on the PCI-express bus.
3. A PCI-express controller and an Ethernet controller, in the case of no Ethernet controller on the PCI-express bus.

Another functionality provided by the digital logic is a programming interface. This module allows for programmable registers to be set by a software application which the user interacts with on a computer. The SATA or PCI-express interfaces allow for register programmability. This programmability is presented to the user as status registers and control registers. The status registers consist of information pertinent for the user, including:

- Information about all local disks connected to the controller.
- Performance of connected cloud drives.
- Array configuration information.
- Array health information.
- Status of yet-to-be-streamed parity data.
- Information about the cloud drives connected to any array.

In addition to status registers, control registers are visible to the user for controller programmability. Control registers consist of parameters the user can set, clear, or otherwise configure to change the functionality of the controller, including

- Configuring encryption and compression parameters.
- Selecting the RAID level and disks.
- Selecting the number of cloud disks.
- Communication controller configuration, e.g. MAC address, IP address, etc.

The digital logic implements compression and encryption algorithms. If the users want to compress data before encryption or before storing it unencrypted, they can, at the cost of computational time and power, save storage space. Additionally, when the users want to store their data encrypted, whether it is stored locally or in the cloud, the encryption core handles the encryption and decryption of data. Typically, data is first compressed, and then encrypted. This is because encryption usually prevents the ability to perform much compression, whereas compressing data first better enables the shrinking of file sizes.

Finally, the digital logic handles write caching. As previously explained, RAID 4 typically incurs a write penalty due to the parity drive becoming a bottleneck for write performance. As discussed in Chapter 3, Lime Technology's unRAID utilizes a similar scheme, combining a write cache with a processor, as in this scheme. However, their method is limited to a single parity disk, which is also a local disk. The proposed hardware-based cloud-RAID controller improves upon this scheme by providing the ability to store parity in the cloud, which provides reliability and security, and also by allowing the use of multiple parity disks, enabling data recovery in the event of complete drive loss. The digital functionality, when combined with the microprocessor functionality explained in the next section, provides many improvements upon existing hardware RAID schemes.

5.5.2 Microprocessor

The microprocessor primarily implements the required functionality to stream data to cloud storage drives as array writes are performed, and the ability to read cloud data to recover an array in the event of failure. The microprocessor also handles logging into cloud storage providers and optimization of data storage locations.

The primary task of the microprocessor is to synchronize data to and from the selected cloud locations. The microprocessor can utilize libraries written in C in order to communicate over HTTP/HTTPS to various cloud storage APIs via the Ethernet controller. The system can optimize the sending of data to the cloud by only updating the cloud data that needs updating. For example, in the case where parity data is stored in the cloud, the processor can, instead of recalculating parity for the entire array whenever data changes, only update the data that needs to be changed.

The processor can also perform location optimization by monitoring which remote cloud connections have the greatest throughput. For example, a user can log into as many cloud-based accounts as desired, and the processor can monitor which sites have the greatest throughput. The processor can ensure that data is always stored to the fastest connection, enabling faster array write speeds. In this scheme, the processor keeps track of a map of which information is stored to which cloud location, so data can end up aggregated across many different cloud storage locations.

The processor can perform system monitoring by collecting information about the system and reporting it. It can generate log files for automatic status reporting over time. It can keep track of system events such as creating, changing, or destroying existing arrays, errors that have occurred, and other pertinent information for the user.

Finally, the processor offers reprogramability in the event that a user needs to interact with a specific API. Included software running on the user's computer can store API implementations for specific vendors. When the user specifies which providers they want to use, the microprocessor can automatically reconfigure and download the needed APIs.

CHAPTER 6. CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

Both software and hardware-based local+cloud-RAID implementations have the ability to significantly increase the performance, reliability, and security of storage systems. Local+cloud-RAID 4 and local+cloud-RAID-DP offer improvements in MTTF, online privacy, read performance, write performance, and superior utilization of local data storage area in comparison to traditional RAID architectures.

The performance enhancements of a local+cloud-RAID 4 system were measured in Chapter 4 with a software implementation. These performance gains include:

1. Read/write performance comparable to RAID 5
2. Performance approaching RAID 0 levels
3. The ability to create a local RAID 0 array while maintaining redundancy through the use of cloud storage, enabling maximum performance of local storage.

The software implementation utilizing a ramdisk which is streamed to the cloud allows for a relatively simple method to enhance performance. With the added ability to asynchronously manage parity in a RAID 4 array, performance can be further enhanced by allowing parity to be calculated in the background, after disk operations have finished. The hardware implementation can further improve on the software implementation by combining all of the benefits of hardware RAID, as outlined in Chapter 5, with the benefits of local+cloud-RAID 4 or local+cloud-RAID-DP.

The reliability enhancements of the software and hardware implementations are many. The architectures provide all of the reliability benefits that redundancy offers. The ability to create a number of dedicated parity drives matching the number of local disks is the paragon of storage reliability; all local disks can simultaneously fail and the original data can still be reconstructed.

Additionally, even though all redundant information can be stored in the cloud, cloud storage can be acted upon by malicious entities, potentially exposing private or otherwise confidential information. By storing parity information rather than a complete set of the original data, privacy can be guaranteed. Furthermore, if multiple cloud storage providers are utilized, then even though complete parity information may be stored on the cloud, a malicious entity has a much more difficult task: compromising *all* of the cloud storage accounts associated with the local+cloud-RAID array. This greatly enhances overall system security. The scheme of storing parity in the cloud provides the following security benefits:

1. Preventing time loss due to failed data.
2. Providing enhanced privacy and confidentiality.
3. Protecting against physical damage to redundant disks by migrating them to the cloud.
4. Protecting against financial loss.
5. Protecting against reputational damage.
6. Protecting against legal issues due to the unwanted disclosure of sensitive information.
7. Protecting against the loss of trade secrets or proprietary information.

It is an assertion of this thesis that no existing redundant storage architecture provides the same levels of benefits to performance, reliability, availability, privacy, and security that local+cloud-RAID with one or more dedicated parity drives achieves. Few RAID architectures purport to enhance system security or privacy. Even putting security and privacy aside, the architecture provides substantial benefit to performance by combining the ability to utilize the fastest RAID level, RAID 0, in conjunction with highly reliable and available cloud storage. In short, the architecture provides:

1. The fastest theoretical performance, because all local storage can be RAID 0.
2. The greatest possible reliability through the use of highly available cloud storage.
3. Guaranteed protection of all user data through multiple dedicated parity drives.
4. Guaranteed privacy in the case a cloud storage location is compromised.

6.2 Future Work

In Chapter 5 I presented a hardware-based local+cloud-RAID architecture, but an actual implementation of this architecture represents a multi-year effort among many engineers, and is beyond the scope of this thesis.

In chapter 4 I utilized mdadm to create a RAID 4 array which included a ramdisk as a dedicated parity drive. However, mdadm can throttle performance in the case of RAID 4 because parity information is calculated in real-time as data is written to the primary array disks. The relaxation of this requirement offers a potential for increased write speeds because parity would not have to be maintained in real time, consuming resources. Parity calculation can occur in the background after write tasks have completed, enabling faster performance, at the small expense of data becoming vulnerable to loss for a short period of time while the parity is being updated.

Another way to implement asynchronous parity generation is through using mdadm or another tool to create RAID 0 arrays, and then gaining access to the lower-level structure of the disks and calculating parity manually across the RAID 0 array, asynchronously to the RAID 0 array itself.

As many drives implement write caching, a system could be implemented that utilizes the write caches as a means to calculate parity.

REFERENCES

[1] D. A. Patterson, G. Gibson, and R. H. Katz, *A case for redundant arrays of inexpensive disks (RAID)*. ACM, 1988, vol. 17, no. 3. vii, 4, 41, 42

[2] A. Bessani, M. Correia, B. Quaresma, F. André, and P. Sousa, "DepSky: Dependable and secure storage in a cloud-of-clouds," in *Proceedings of the Sixth Conference on Computer Systems*, ser. EuroSys '11. New York, NY, USA: ACM, 2011, pp. 31–46. [Online]. Available: http://doi.acm.org/10.1145/1966445.1966449 1, 2

[3] H. Takabi, J. B. D. Joshi, and G. J. Ahn, "Security and privacy challenges in cloud computing environments," *IEEE Security Privacy*, vol. 8, no. 6, pp. 24–31, Nov 2010. 1

[4] C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring data storage security in cloud computing," in *Quality of Service, 2009. IWQoS. 17th International Workshop on*, July 2009, pp. 1–9. 1, 2

[5] M. A. Alzain, B. Soh, and E. Pardede, "MCDB: Using multi-clouds to ensure security in cloud computing," in *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on*, Dec 2011, pp. 784–791. 1, 2

[6] H. Abu-Libdeh, L. Princehouse, and H. Weatherspoon, "RACS: A case for cloud storage diversity," in *Proceedings of the 1st ACM Symposium on Cloud Computing*, ser. SoCC '10. New York, NY, USA: ACM, 2010, pp. 229–240. [Online]. Available: http://doi.acm.org/10.1145/1807128.1807165 1

[7] M. Schnjakin, D. Korsch, M. Schoenberg, and C. Meinel, "Implementation of a secure and reliable storage above the untrusted clouds," in *Computer Science Education (ICCSE), 2013 8th International Conference on*, April 2013, pp. 347–353. 1, 2, 31

[8] M. D. Francesco, N. Li, M. Raj, and S. K. Das, "A storage infrastructure for heterogeneous and multimedia data in the Internet of Things," in *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*, Nov 2012, pp. 26–33. 1

[9] M. A. Hail, M. Amadeo, A. Molinaro, and S. Fischer, "Caching in named data networking for the wireless Internet of Things," in *Recent Advances in Internet of Things (RIoT), 2015 International Conference on*, April 2015, pp. 1–6. 1

[10] M. Fazio, A. Celesti, M. Villari, and A. Puliafito, "The need of a hybrid storage approach for IoT in PaaS cloud federation," in *Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on*, May 2014, pp. 779–784. 1

[11] C. Cecchinel, M. Jimenez, S. Mosser, and M. Riveill, "An architecture to support the collection of big data in the Internet of Things," in *Services (SERVICES), 2014 IEEE World Congress on*, June 2014, pp. 442–449. 1

[12] H. Graupner, K. Torkura, P. Berger, C. Meinel, and M. Schnjakin, "Secure access control for multi-cloud resources," in *Local Computer Networks Conference Workshops (LCN Workshops), 2015 IEEE 40th*, Oct 2015, pp. 722–729. 1, 2

[13] Y. Chi, W. Cai, Z. Hong, H. C. B. Chan, and V. C. M. Leung, "A privacy and price-aware intercloud system," in *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, Nov 2015, pp. 298–305. 1, 2

[14] C. W. Ling and A. Datta, "InterCloud RAIDer: A do-it-yourself multi-cloud private data backup system," in *Proceedings of the 15th International Conference on Distributed Computing and Networking - Volume 8314*, ser. ICDCN 2014. New York, NY, USA: Springer-Verlag New York, Inc., 2014, pp. 453–468. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-45249-9_30 1, 2

[15] K. D. Bowers, A. Juels, and A. Oprea, "HAIL: A high-availability and integrity layer for cloud storage," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, ser. CCS '09. New York, NY, USA: ACM, 2009, pp. 187–198. [Online]. Available: http://doi.acm.org/10.1145/1653662.1653686 1, 2, 35

[16] G. Chockler, R. Guerraoui, I. Keidar, and M. Vukolic, "Reliable distributed storage," *Computer*, vol. 42, no. 4, pp. 60–67, April 2009. 1, 2, 35

[17] M. Schnjakin, R. Alnemr, and C. Meinel, *Web Information Systems Engineering – WISE 2010 Workshops: WISE 2010 International Symposium WISS, and International Workshops CISE, MBC, Hong Kong, China, December 12-14, 2010, Revised Selected Papers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, ch. A Security and High-Availability Layer for Cloud Storage, pp. 449–462. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-24396-7_36 1, 31

[18] B. Mao, S. Wu, and H. Jiang, "Improving storage availability in cloud-of-clouds with hybrid redundant data distribution," in *Parallel and Distributed Processing Symposium (IPDPS), 2015 IEEE International*, May 2015, pp. 633–642. 1, 2, 35

[19] Q. Zhang, S. Li, Z. Li, Y. Xing, Z. Yang, and Y. Dai, "CHARM: A cost-efficient multi-cloud data hosting scheme with high availability," *IEEE Transactions on Cloud Computing*, vol. 3, no. 3, pp. 372–386, July 2015. 1, 2, 31, 35

[20] M. Schnjakin, T. Metzke, and C. Meinel, "Applying erasure codes for fault tolerance in cloud-RAID," in *Computational Science and Engineering (CSE), 2013 IEEE 16th International Conference on*, Dec 2013, pp. 66–75. 2

[21] M. Vrable, S. Savage, and G. M. Voelker, "BlueSky: A cloud-backed file system for the enterprise," in *Proceedings of the 10th USENIX Conference on File and Storage Technologies*, ser. FAST'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 19–19. [Online]. Available: http://dl.acm.org/citation.cfm?id=2208461.2208480 2

[22] C. Selvakumar, G. J. Rathanam, and M. R. Sumalatha, "PDDS - improving cloud data storage security using data partitioning technique," in *Advance Computing Conference (IACC), 2013 IEEE 3rd International*, Feb 2013, pp. 7–11. 2

[23] M. Schnjakin and C. Meinel, "Evaluation of cloud-RAID: A secure and reliable storage above the clouds," in *Computer Communications and Networks (ICCCN), 2013 22nd International Conference on*, July 2013, pp. 1–9. 2, 28, 31, 35

[24] ——, "Scrutinizing the state of cloud storage with cloud-RAID: A secure and reliable storage above the clouds," in *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on*, June 2013, pp. 309–318. 2, 35

[25] G. Song, S. Kim, and D. Seo, "Saveme: client-side aggregation of cloud storage," *IEEE Transactions on Consumer Electronics*, vol. 61, no. 3, pp. 302–310, Aug 2015. 2, 35

[26] B. Mao, S. Wu, and H. Jiang, "Exploiting workload characteristics and service diversity to improve the availability of cloud storage systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. PP, no. 99, pp. 1–1, 2015. 2, 35

[27] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proceedings of the 4th International Conference on Security and Privacy in Communication Networks*, ser. SecureComm '08. New York, NY, USA: ACM, 2008, pp. 9:1–9:10. [Online]. Available: http://doi.acm.org/10.1145/1460877.1460889 2

[28] C. W. Chang, P. Liu, and J. J. Wu, "Probability-based cloud storage providers selection algorithms with maximum availability," in *Parallel Processing (ICPP), 2012 41st International Conference on*, Sept 2012, pp. 199–208. 2

[29] D. Petcu, "Consuming resources and services from multiple clouds," *J. Grid Comput.*, vol. 12, no. 2, pp. 321–345, Jun. 2014. [Online]. Available: http://dx.doi.org/10.1007/s10723-013-9290-3 2

[30] "Standard RAID levels," https://en.wikipedia.org/wiki/Standard_RAID_levels, accessed: 2016-10-10. 5, 6, 8, 9, 10, 11, 12, 13, 15

[31] M. L. Shooman, *Reliability of Computer Systems and Networks: Fault Tolerance, Analysis and Design*. New York, USA: Wiley-Interscience, 2002, pp. 112–114,438–441. 5, 6, 7, 8, 9, 10, 12, 22, 23, 24, 25, 36

[32] "What is RAID 0," http://blog.open-e.com/what-is-raid-0/, accessed: 2016-03-21. 5

[33] "Cisco UCS servers RAID guide, chapter 1: RAID overview," http://www.cisco.com/c/en/us/td/docs/unified_computing/ucs/c/sw/raid/configuration/guide/RAID_GUIDE/IntroToRAID.pdf#page=14, accessed: 2016-03-22. 5, 6, 7, 8, 9, 10, 12, 13, 14

[34] "What are RAID 1, RAID 1+0 and RAID 0+1," http://blog.open-e.com/what-are-raid-1-raid-10-and-raid-01/, accessed: 2016-03-21. 6, 12

[35] "RAID 2, RAID 3, RAID 4 what it is, how it works? the history lesson," http://blog.open-e.com/raid-2-raid-3-raid-4-what-it-is-how-it-works-the-history-lesson/, accessed: 2016-03-21. 7, 8

[36] "RAID 5? RAID 6? or other alternativee?" http://blog.open-e.com/raid-5-raid-6-or-other-alternativee/, accessed: 2016-03-21. 9, 10

[37] "The mathematics of RAID-6," https://www.kernel.org/pub/linux/kernel/people/hpa/raid6.pdf, accessed: 2016-10-11. 10

[38] "Nested RAID levels," https://en.wikipedia.org/wiki/Nested_RAID_levels, accessed: 2016-10-10. 14, 15, 16

[39] "Understanding and using RAID 100," http://www.storagecraft.com/blog/using\discretionary{-}{}{}raid\discretionary{-}{}{}100/, accessed: 2016-10-11. 14

[40] "Understanding RAID performance at various levels," http://www.storagecraft.com/blog/raid\discretionary{-}{}{}performance/, accessed: 2016-10-10. 15

[41] "What is the hadoop distributed file system (HDFS)?" http://www-01.ibm.com/software/data/infosphere/hadoop/hdfs/, accessed: 2016-10-11. 16

[42] "Welcome to apache hadoop!" http://hadoop.apache.org/, accessed: 2016-10-11. 16

[43] "BeeGFS," http://www.beegfs.com/content/documentation/, accessed: 2016-10-11. 17

[44] "About moosefs," https://moosefs.com/products.html, accessed: 2016-10-11. 17

[45] "Red hat snatches storage gluster file system for $136m," http://www.theregister.co.uk/2011/10/04/redhat_buys_gluster/, accessed: 2016-10-11. 17

[46] "Contributors," https://btrfs.wiki.kernel.org/index.php/Contributors, accessed: 2016-10-11. 17

[47] "Main page," https://btrfs.wiki.kernel.org/index.php/Main_Page, accessed: 2016-10-11. 17

[48] "Zfs," https://en.wikipedia.org/wiki/ZFS, accessed: 2016-10-11. 18

[49] "Zfs," http://www.freenas.org/zfs/, accessed: 2016-10-11. 18

[50] "mdadm," http://neil.brown.name/blog/mdadm, accessed: 2016-10-11. 18

[51] "Linux raid," https://raid.wiki.kernel.org/index.php/Linux_Raid, accessed: 2016-10-11. 18

[52] D. Colarelli, D. Grunwald, and M. Neufeld, "The case for massive arrays of idle disks (maid)," in *The 2002 Conference on File and Storage Technologies*, 2002. 19

[53] "Non-standard raid levels primer: Raid 1e," http://www.techrepublic.com/article/non-standard-raid-levels-primer-raid-1e/, accessed: 2016-10-11. 19, 20

[54] "Zfs and raid-z recoverability and performance," quetek.com/zfsandraidz.htm, accessed: 2016-10-11. 20

[55] "9 alternatives for windows home servers drive extender," http://www.howtogeek.com/howto/36458/9-alternatives-for-windows-home-servers-drive-extender/, accessed: 2016-10-11. 20

[56] "Drive bender public release arriving this week," http://www.wegotserved.com/2011/10/10/drive-bender-public-release-arriving-week/, accessed: 2016-10-11. 20

[57] "Stablebit drivepool 2 year review," http://www.homemediatech.net/stablebit-drivepool-2-year-review/83, accessed: 2016-10-11. 20

[58] "Intel rapid storage technology," http://www.intel.com/content/www/us/en/architecture-and-technology/rapid-storage-technology.html, accessed: 2016-10-11. 21

[59] "IBM spectrum scale," http://www-03.ibm.com/systems/storage/spectrum/scale/, accessed: 2016-10-11. 21

[60] "Network-attached storage," https://lime-technology.com/network-attached-storage/, accessed: 2016-10-11. 22, 33

[61] "Beyondraid," http://www.drobo.com/drobo/beyondraid/, accessed: 2016-10-11. 22

[62] "RAID level 7," http://www.pcguide.com/ref/hdd/perf/raid/levels/singleLevel7-c.html, accessed: 2016-10-11. 22

[63] C. Hansen and J. Archibald, "The case for raid 4: Cloud-raid integration with local storage," in *2016 International Conference on Availability, Reliability and Security*, to be published. 22, 28, 35

[64] C. Herley, D. Florencio, and P. van Oorschot, "An administrators guide to internet password research," November 2014. [Online]. Available: https://www.microsoft.com/en-us/research/publication/an-administrators-guide-to-internet-password-research/ 29

[65] G. Tajadod, L. Batten, and K. Govinda, "Microsoft and Amazon: A comparison of approaches to cloud security," in *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, Dec 2012, pp. 539–544. 30, 36

[66] "Introducing geo-replication for Windows Azure storage," https://blogs.msdn.microsoft.com/windowsazurestorage/2011/09/15/introducing-geo-replication-for-windows-azure-storage/, accessed: 2016-03-20. 30, 36

[67] "Amazon SimpleDB FAQs," http://aws.amazon.com/simpledb/faqs/, accessed: 2016-03-20. 30, 36

[68] "Amazon EC2 Service Level Agreement," http://aws.amazon.com/ec2/sla/, accessed: 2016-03-20. 30, 36

[69] "Service level agreements," https://azure.microsoft.com/en-us/support/legal/sla/, accessed: 2016-03-20. 30, 36

[70] J. Araujo, P. Maciel, M. Torquato, G. Callou, and E. Andrade, "Availability evaluation of digital library cloud services," in *Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on*, June 2014, pp. 666–671. 30

[71] "Summary of the Amazon EC2 and Amazon RDS service disruption in the US east region," http://aws.amazon.com/message/65648/, accessed: 2016-03-22. 31

[72] “Amazon’s cloud crash disaster permanently destroyed many customers’ data,” http://www.businessinsider.com/amazon-lost-data-2011-4, accessed: 2016-03-22. 31

[73] S. Gregori, A. Cabrini, O. Khouri, and G. Torelli, “On-chip error correcting techniques for new-generation flash memories,” *Proceedings of the IEEE*, vol. 91, no. 4, pp. 602–616, April 2003. 33

[74] H. Kaneko, T. Matsuzaka, and E. Fujiwara, “Three-level error control coding for dependable solid-state drives,” in *Dependable Computing, 2008. PRDC ’08. 14th IEEE Pacific Rim International Symposium on*, Dec 2008, pp. 281–288. 33

[75] R. Bez, E. Camerlenghi, A. Modelli, and A. Visconti, “Introduction to flash memory,” *Proceedings of the IEEE*, vol. 91, no. 4, pp. 489–502, April 2003. 33

[76] “Hardware raid vs. software raid: Which implementation is best for my application?” http://www.adaptec.com/nr/rdonlyres/14b2fd84-f7a0-4ac5-a07a-214123ea3dd6/0/4423_sw_hwraid_10.pdf, accessed: 2016-10-16. 51

[77] “Search the largest api directory on the web,” http://www.programmableweb.com/category/storage/apis?category=20171, accessed: 2016-10-23. 53

[78] “Other http/ftp client libraries,” https://curl.haxx.se/libcurl/competitors.html, accessed: 2016-10-23. 53