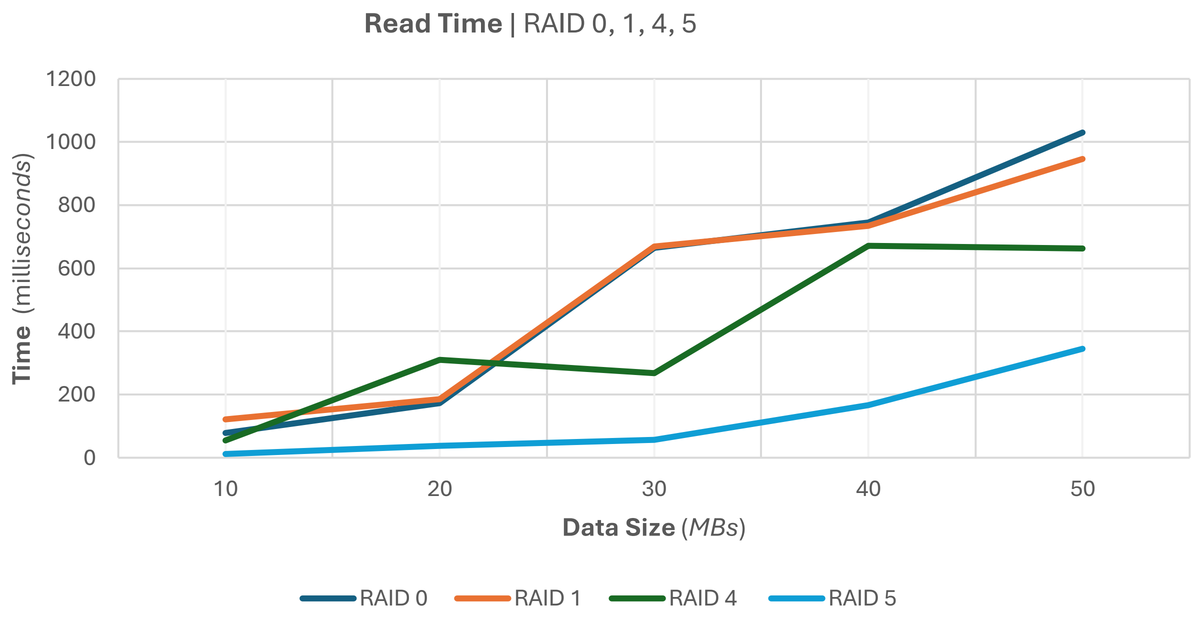
Dalton Driscoll
Shem Louisy
Abumere Okhihan

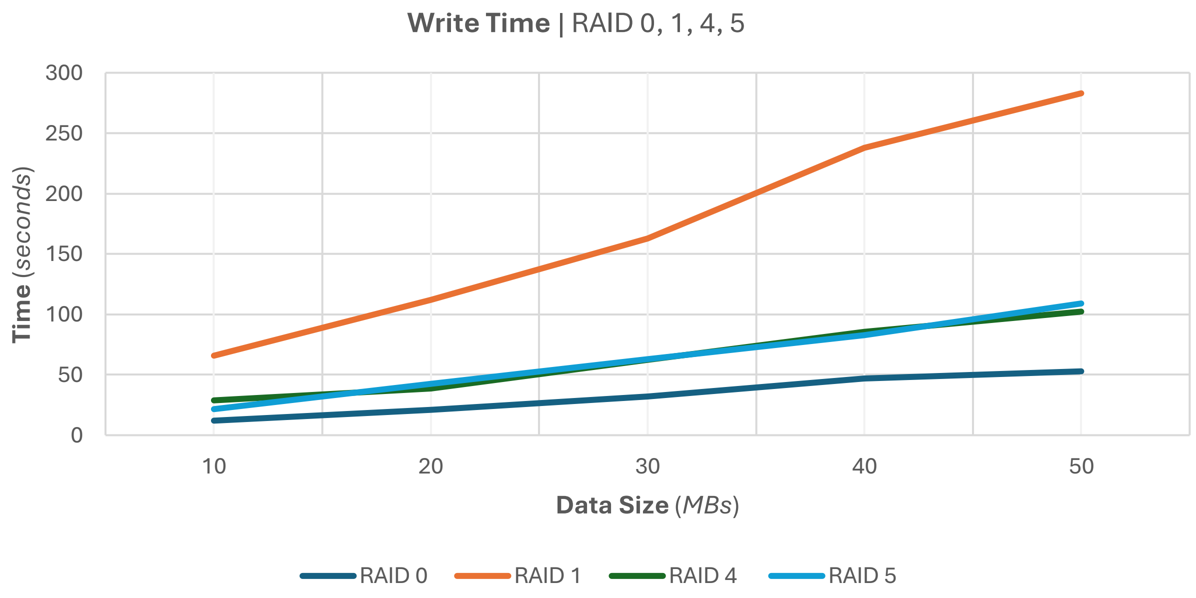
Wednesday, April 9th, 2025

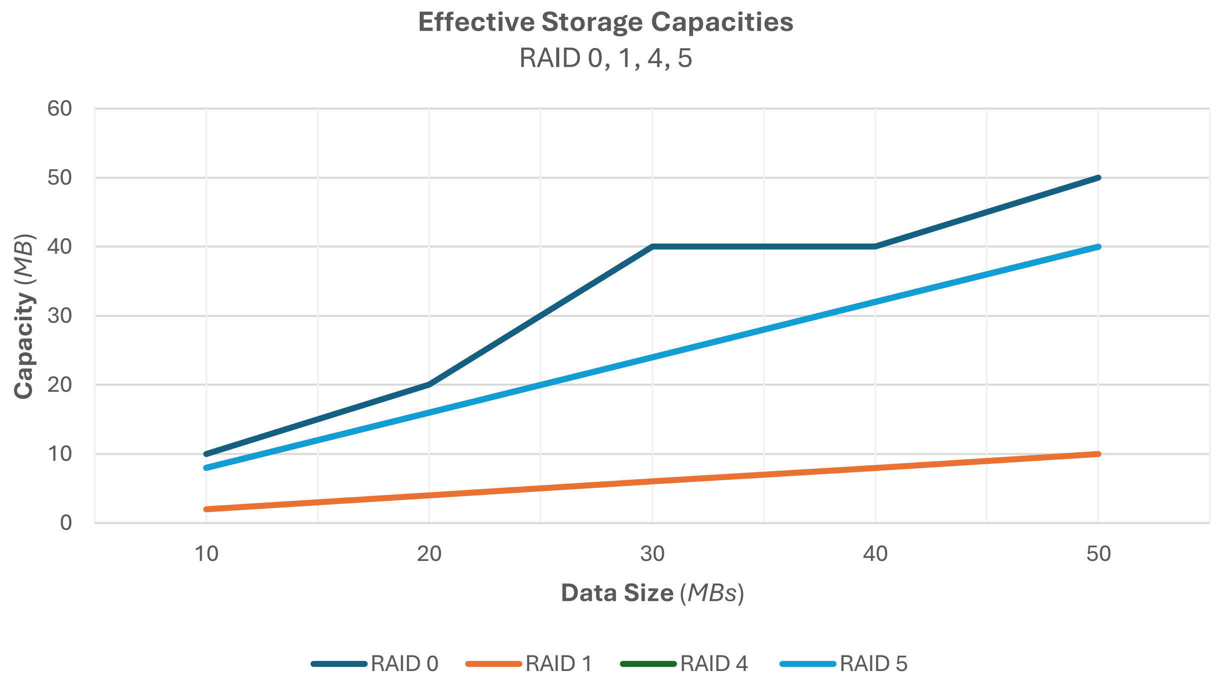
EECE 4811/5811 Operating Systems Spring 2025

Professor Tseng

Homework 7







The results demonstrate the varying trade-offs between RAID 0, RAID 1, RAID 4, and RAID 5 in terms of performance and storage efficiency. The first graph highlights the read times, showing that RAID 0 consistently delivers the fastest performance, while RAID 1 has the highest read times due to its mirroring process. RAID 5 and RAID 4 fall somewhere in between, with RAID 5 generally offering better efficiency. The second graph focuses on write times, where RAID 1 again shows the slowest performance, likely due to the need for duplicate data writes. RAID 0 is the fastest, while RAID 4 and RAID 5 provide a middle ground by incorporating parity-based redundancy without the full duplication of RAID 1. These variations underscore the importance of selecting a RAID configuration that aligns with the need for speed or fault tolerance, as RAID 0 prioritizes speed but lacks redundancy, whereas RAID 5 provides a balance between performance and data protection.

The third graph provides insight into effective storage capacities, showing that RAID 0 and RAID 5 make the most efficient use of storage space, while RAID 1 offers the least capacity due to its mirroring process. RAID 4 falls in between, as it uses dedicated parity for redundancy but doesn't require full duplication. This highlights how different RAID levels impact usable storage, and the trade-offs involved in ensuring data protection. For users or organizations making storage decisions, these results reinforce the idea that RAID 0 is best for speed-driven applications where redundancy isn't critical, RAID 1 is ideal for situations requiring robust data protection, and RAID 5 presents a strong compromise between performance and reliability.

The results generally align with expected RAID performance characteristics. RAID 0 excels in speed because it stripes data across multiple drives without any redundancy, leading to quick read and write times. RAID 1, which mirrors all data, naturally has slower write speeds due to duplication, while read speeds can be improved depending on the implementation. RAID 4 and RAID 5 both use parity for fault tolerance, but RAID 5 usually performs better due to its distributed parity, which avoids the bottleneck of a single dedicated parity drive in RAID 4. The effective storage capacities also make sense, with RAID 0 offering full capacity, RAID 1 reducing available space due to mirroring, and RAID 5 providing more efficient use of storage compared to RAID 1.

Code Description

This Go program simulates the functionality of four RAID levels, RAID 0, 1, 4, and 5 using plain files to act like physical disks. The core idea is to mimic how data would be distributed across disks in different RAID setups, such as striping in RAID 0, full mirroring in RAID 1, and parity-based fault tolerance in RAID 4 and 5. Each "disk" is just a file, and operations like reading and writing are done at block-level granularity, with a fixed 4KB block size. There's a shared interface RAID for all implementations, keeping things modular and consistent.

The benchmark section generates a set amount of random data (50 MB default, that can be changed to simulate workloads), writes it block-by-block to each RAID configuration, and then reads it back, measuring how long both operations take. This helps visualize performance trade-offs across the RAID types. At the end, it also calculates the effective storage capacity for each RAID setup, showing how redundancy affects usable space.

Code Use

Navigate to the folder with the "`raid.go`" file

In a terminal run the go file by typing "`go run raid.go`"

**The go.mod file must be in the folder in order for the program to run*

References

[1] S. Bhargava, "RAID Explained | Redundant Array of Independent Disks," YouTube, Apr. 8, 2022. https://www.youtube.com/watch?v=U-OCdTeZLac

[2] P. Chen, E. Lee, G. Gibson, R. Katz, and D. Patterson, "RAID: High-performance, reliable secondary storage," ACM Computing Surveys, vol. 26, no. 2, pp. 145–185, Jun. 1994.

[3] T. Anderson and M. Dahlin, “Operating Systems: Principles and Practice,” 2nd ed. Recursive Books, 2014. https://ospp.cs.washington.edu/

[4] M. Kerrisk, The Linux Programming Interface: A Linux and UNIX System Programming Handbook, 1st ed. No Starch Press, 2010.

[5] A. Donovan and B. Kernighan, The Go Programming Language, 1st ed. Addison-Wesley, 2015.