**Samonorom Chum**

**Thursday May 8, 2025**

**EECE 4811/5811 Operating System Spring 2025**

**Professor Tseng**

**HW7**

1. **RAID Simulation in Go (6 pt)**

* **Overview**: You will simulate various RAID levels using Go. Rather than working with actual hardware, you will use regular files to represent physical disks. This approach allows you to focus on the logic of RAID levels and understand the trade-offs between performance and redundancy.
* **Objectives/Concepts**: By the end of this assignment, you should be able to:
* Understand the key differences and trade-offs between RAID 0, 1, 4, and 5.
* Implement data striping, mirroring, and parity logic.
* Simulate physical disks using files.
* Measure and analyze read/write performance of different RAID levels.
* **Specification of your RAID implementation:**
* Use **five files** **(e.g., disk0.dat to disk4.dat)** to simulate physical disks.
* Each disk file should support **basic read and write** operations at specified block offsets.
* **Define a block size** **(e.g., 512 bytes or 4 KB)** and maintain a logical mapping from logical blocks to physical locations.
* You should implement **RAID 0, 1, 4, and 5 – use XOR for RAID 5**
* Interface – Each RAID level should implement this interface.

**type RAID interface {**

**Write(blockNum int, data []byte) error**

**Read(blockNum int) ([]byte, error)**

**}**

* Note: Ensure files are **flushed (fsync)** properly to simulate real disk write delays.
* Note1: you don’t need to worry about edge cases, such as corrupted files, disk failures, read/write beyond bounds.
* **Benchmarking and Evaluation**:
* Write a benchmark tool that:

1. Writes a large chunk of data (e.g., 100 MB) in blocks.
2. Reads back the data in blocks.
3. Measures and reports the time taken for each operation (total and per block).
4. Run the benchmark for all four RAID levels using identical data sizes and workload

* Plot or print a summary comparing, under different workload and configuration:

1. Write performance
2. Read performance
3. Effective storage capacity

* Compare the performance numbers you have and the RAID analysis in the textbook. Does the trend match? If not, briefly identify potential reasons.

# HW7 Report:

# Project Structure:

* The directory structure is as follows:

hw7\_raid\_sim/

├── .gitignore

├── go.mod

├── main.go (contains benchmark function).

├── data/

├── disk0.dat

├── disk1.dat

├── disk4.dat

├── disk5.dat

├── disk/

└── disk.go

├── raid/

├── interface.go

├── raid0.go

├── raid1.go

├── raid4.go

├── raid5.go

└── raid\_test.go

└── utils/

└── timer.go

# Files/Folder Explanation:

* 1. The main folder was in my home direction then “hw7\_raid\_sim/”.
  2. Go.mod – allows us to run go within the folder.
  3. Main.go – holds the benchmark function and flushes the disk data files.
     1. Define block size is set to 4KB (4096 bytes).
  4. The “data” folder contains disk 0, 1, 4, and 5 to represent the parity disks (disk0.dat to disk4.dat).
  5. The “disk” folder contains “disk.go“ which is the read/write from the disk in the data folder.
  6. The actual raid levels are in the raid folder (raid0.go, raid1.go, raid4.go, and raid5.go.
     1. Additionally, the “interface.go” holds the required interface function.

# Running the File:

* Open Ubuntu (I used WSL and Ubuntu 24.04) and go to the “hw7\_raid\_sim/” folder.
* Inside the folder “hw7\_raid\_sim”
* Enter the command “go run . –level=raid[Chosen #] --disks=[Chosen #] –size=[Choosen #]”
  + For raid, you can replace the “[chosen #] next to “raid” with a 0, 1, 4, or 5.
    - Each number responds to which RAID level you want.
  + For disks, you can replace the “[chosen #] next to “disks” for the number of disks you want.
    - RAID1 requires at least 2 disks.
    - RAID4 and RAID5 require at least 3 disks.
  + For size, you can replace the “[chosen #] next to “size” for the size you want in MB.
    - I used 10MB as initial testing, then 100MB as required.
  + **Example of how to run the code with flags:**

**“go run . --level=raid5 --disks=5 --size=100”**

# RAID Implementations:

The project supports the following RAID levels:

* RAID 0: Striping without parity or redundancy.
* RAID 1: Mirroring across all disks.
* RAID 4: Block-level striping with a dedicated parity disk.
* RAID 5: Block-level striping with distributed parity.

# Benchmarking Tool

The benchmark tool writes and reads a user-defined amount of data to each RAID level. It measures the total time taken for writes and reads, helping assess the performance of each implementation.

## Test run:

I tested each RAID level (0, 1, 4, and 5) with a disk count of 4 and size of 100MB.

* go run . --level=raid0 --disks=4 --size=100
* go run . --level=raid1 --disks=4 --size=100
* go run . --level=raid4 --disks=4 --size=100
* go run . --level=raid5 --disks=4 --size=100

# Results:

* “Plot or print a summary comparing, under different workload and configuration:”
  + Write performance
  + Read performance
  + Effective storage capacity
* **Answer**: When parity is involved, it is understandably significantly slower. RAID4 and RAID5 are still not as fast as RAID0 as expected, but it is more efficient than RAID1’s “1:1 copy”.
* “Compare the performance numbers you have and the RAID analysis in the textbook. Does the trend match? If not, briefly identify potential reasons?”
* **Answer:** RAID 0,4,5 generally matches overall with the textbook value, but RAID1 is significantly slower possibly due to the use of a flush.

**RAID Level: raid0**

Disks: 4 | Size: 100 MB | Blocks: 25600

Total Write Time: 52.842025s

Total Read Time: 203.957ms

**RAID Level: raid1**

Disks: 4 | Size: 100 MB | Blocks: 25600

Total Write Time: 3m29.330148s

Total Read Time: 243.114ms

**RAID Level: raid4**

Disks: 4 | Size: 100 MB | Blocks: 25600

Total Write Time: 1m39.643119s

Total Read Time: 160.748ms

**RAID Level: raid5**

Disks: 4 | Size: 100 MB | Blocks: 25600

Total Write Time: 1m28.61043s

Total Read Time: 142.341ms

**References:**

1. ChatGPT, “Assistance with HW7 Raid Simulation in Go,” OpenAI, personal communication, May. 7, 2025.

**Prompts Used with ChatGPT:**

* 1. Uploaded HW7 in ChatGPT, and used the prompt “Help me get started on HW7”
  2. Prompt ChatGPT to help uninstall my old Ubuntu and WSL software, and install WSL and Ubuntu 24.04.
  3. I then prompted ChatGPT to “help me download gedit, Go, and other needed software inside Ubuntu”.
  4. I then prompted ChatGPT to “help me use VS Code within Ubuntu”.
  5. I prompted ChatGPT to “help me get started again”, which it helped me develop skeleton codes.
  6. “Where should I make the folder?”
  7. “I have VS Code open, help me set up the folder and files”
  8. “Show me how `main.go` is supposed to look”
  9. “Show me the file structure”
  10. I then used a number of “yes” to get ChatGPT to help me program each .go file”
  11. “Show me the full code scaffold and check if it matches HW7 requirements”
  12. “Can you make `main.go` the benchmark file instead
  13. “Remove the CSV logging option”
  14. “Give me a test matrix for different RAID levels and disk counts”
  15. “Can I run all tests in a script in `main.go?”
  16. “Go back to the previous single-run version”
  17. “Check if files match HW7 spec”
  18. “Is this version of `disk.go` correct?”
  19. “Change block size from 512 to 4KB”
  20. “Check if other files still use 512 bytes”
  21. “`raid0.go` is bugging out”
  22. “Fix and update `raid0.go` with padding”
  23. “Why does `blockSize` give an error when used directly?”
  24. “Generate a Word report”