# CS 537 - Introduction to Operating Systems

## Project 4: Software RAID


**Aman Rakesh Chadha**
**(9068354597)**

# 1 Objectives

The goals for this project are:

- Develop better understanding of the specifics of different RAID implementations
- Analyze tradeoffs in RAID write strategies
- Understand data recovery strategies

# 2 Overview

In this system you will implement a software RAID manager implementing RAID 0 (striping), RAID 1 (mirroring), RAID 4 (fixed parity disk), and RAID 5 (floating parity), and RAID 10 (mirroring + striping). For each strategy, you will implement a RAID manager capable of normal mode operation (no failures), failed-mode operation (working with one failed disk), and recovery (restoring normal mode with a new disk).

Rather than assign each of you a set of real disks, we will instead work with virtual disks stored in ordinary files. Thus, we won't be able to accurately measure the performance impact of different RAID systems. Instead, we will simply count the number of accesses per disk and use that as a proxy for performance.

# 3 Details

You will create a simulator program named raidsim.

# 4 Virtual disk array

Please use the functions defined in disk-array.h. The file test.c shows how to use the functions. The block size for the project is defined in disk.h.

**RAID 0**

Blocks should be striped across disks, starting with the lowest numbered disk and increasing. Each disk gets a **strip** of blocks, which is one or more blocks. For example, with 3 disks and strip size of 2, disk 0 gets blocks 0-1 and 6-7, disk 1 gets blocks 2-3 and 8-9, and disk 3 gets blocks 4-5 and 10-11.

**RAID 10**

Blocks should be mirrored between pairs of disks and then striped across pairs of disks in strip units. Mirrors should be disk i and i+1. For example, with 4 disks and strip size 2, disks 0 and 1 get blocks 0-1 and 4-5, and disks 2 and 3 gets blocks 2-3 and 6-7.

**RAID 4**

Blocks should be striped in strip size units across the first n-1 disks, and disk n should receive the parity. For 4 disks and strip size 3, disk 0 gets blocks 0-2 and 9-11, disk 1 gets

blocks 3-5 and 12-14, disk 2 gets blocks 6-8 and 15-17. The parity disk gets the byte-wise XOR of the n-1 data disks. In this case, block 0 XOR block 3 XOR block 6, block 1 XOR block 4 XOR block 7, block 2 XOR block 5 XOR block 8 etc.

For writes less than a full stripe, you can use additive (read the old blocks and compute new parity) or subtractive (read the old block and old parity) parity. For writes to a full stripe, just write out the data and new parity.

**RAID 5**

Blocks should be striped in strip size units across N disks, reserving one disk for parity. For the first strip, disk 0 should be used for parity, for the second disk 1 etc. For stripe, parity returns to disk 0. For example, with strip size 1 and 4 disks

| Disk 0 | Disk 1 | Disk 2 | Disk 3 |
|---|---|---|---|
| B0 XOR B1 XOR B2 | B0 | B1 | B2 |
| B3 | B3 XOR B4 XOR B5 | B4 | B5 |
| B6 | B7 | B6 XOR B6 XOR B8 | B8 |
| B9 | B10 | B11 | B9 XOR B10 XOR B11 |
| B13 XOR B14 XOR B15 | B13 | B14 | B15 |

For writes less than a full stripe, you can use additive (read the old blocks and compute new parity) or subtractive (read the old block and old parity) parity. For writes to a full stripe, just write out the data and new parity.

# 5 Specification

**Command line**

Command line parameters (these can come in any order):

- -level [0|10|4|5] specifies the RAID level as 0, 10, 4, or 5
- -strip n specifies the number of blocks in a strip.
- -disks n specifies the use of n disks. Some RAID levels may require an even number of disks (RAID 10) or more than 2 disks (RAID 4 and 5)
- -size n specifies the number of blocks per disk
- -trace filename specifies the name a file containing a trace of requests
- -verbose set the global variable "verbose" to 1 (defaults to 0).

All parameters are mandatory except "verbose". A sample command line:
raidsim -level 10 -strip 5 -disks 10 -size 10000 -trace input.txt

**Trace file format**

The trace file contains a sequence of lines with one command per line. There will be exactly one space between each word in a command. The commands can be:

- READ LBA SIZE - read starting at block LBA for SIZE blocks. Prints out the first 4 byte value in each block, **separated by spaces on a single line**.
- WRITE LBA SIZE VALUE - write the 4-byte pattern VALUE repeatedly starting at block LBA for SIZE blocks.
- FAIL DISK - make disk number DISK fail, so it cannot be read or written any more. No output
- RECOVER DISK - make disk number DISK recover as a clean, empty disk that can be read or written. No output
- END - last command in trace

For each RAID type, your simulator should try to complete as much of a command as possible: return as much data as possible on a read and write as much data as possible on a write.

# 6 Output

Your simulator should print the command line from the trace and then execute it, then print the desired output on the following line. After the END command, your simulator should call disk_array_print_stats() to print the statistics for the disks. On read errors, replace the value (which may be one of many in a set of blocks) with the word **ERROR**. Thus, a read may return a mix of values and an error. On write errors, print **ERROR** if any blocks of the write cannot be written.