Fault Tolerance

Introduction

1. Cover reliability, availability, and handling failures gracefully

Dependability

1. Dependability: Quality of delivered service that justifies relying on the system to provide that service
    - Specified service: What behavior should be
    - Delivered service: Actual behavior
    - Dependability means does the delivered match the specified service?
2. System has components (modules)
    - Processor, memory, storage, etc.
    - Each module has an ideal specified behavior
    - Real modules don't always exhibit the ideal behavior

Faults, Errors, and Failures

1. Fault: Module deviates from specified behavior
2. Error: Actual behavior within system differs from specified
3. Failure: System deviates from specified behavior

Fault, Error, and Failure Example

1. Fault example: Programming mistake
    - Add function that works fine, except it returns $5 + 3 = 7$
    - Latent error: Not an error until a specific case occurs
2. Error example: Fault has been activated, causing an effective error
    - We call add with 5 and 3, get 7 and put it in some variable
3. Failure example: Deviation in system behavior
    - Schedule a meeting for 7 AM instead of 8 AM
4. Need a fault for an error to occur, but not all faults cause errors
5. Can have an error that doesn't result in a failure
    - If the value 7 still allows for normal execution, there's no failure
        - if(ADD(5,3) > 0) ...

Laptop Falls Down Quiz

1. Laptop falls out of my bag
2. Hits pavement
3. Pavement develops a crack
4. Crack expands during winter
5. Pavement breaks
6. Pavement needs to be replaced
7. What is the fault, error, and failure for the pavement?
    - Fault: 2
    - Error: 3
    - Failure: 5

Reliability - Availability

1. Reliability: Measures continuous service accomplishment
    - Consider system in one of two states
        - Service accomplishment

    - Service interruption
  - MTTF: Mean time to failure
    - How long do we have service accomplishment before service interruption occurs?
2. Availability: Measures service accomplishment as a fraction of overall time
  - If a system is in each state half of the time, availability is 50%
  - MTTR: Mean time to repair
    - How long does service interruption last until the system returns to the service accomplishment state?
  - Availability = MTTF / (MTTF + MTTR)

Reliability and Availability Quiz

1. Consider a hard disk with the following schedule:
  - Works fine for 12 months
  - Breaks (can't spin), takes 1 month to replace motor
  - Works fine for 4 months
  - Breaks (can't move heads), takes 2 months to repair
  - Works fine for 14 months
  - Breaks (head broken), would take 3 months to fix
  - Throw away, buy new disk
2. What is the MTTF, MTTR, and availability?
  - MTTF: (12 + 4 + 14) / 3 = 10
  - MTTR: (1 + 2 + 3) / 3 = 2
  - Availability = 10 / (10 + 2) = 83.3%

Kinds of Faults

1. By cause
  - Hardware faults: Hardware fails to perform as designed
  - Design faults: Software bugs, hardware design mistakes
  - Operation faults: Operator and user mistakes
  - Environmental faults: Fire, power failure, sabotage, etc.
2. By duration
  - Permanent faults: Once we have it, it doesn't get corrected
    - Wanted to see what's inside a processor and now it's in four pieces
  - Intermittent faults: Last for some duration, but recurring
    - Overclock -> Works fine the crashes
  - Transient faults: Fault last for a while then goes away
    - Alpha particle hits chip causing problems, but go awway after a reboot

Fault Classification Quiz

1. Phone gets wet, heats up, then explodes
  - Phone getting wet is a transient fault by duration
  - Phone getting wet is an environmental fault by cause
2. Phone was supposed to prevent itself from operating when wet
  - Heating up is a design fault by cause
  - Heating up is a permanent fault by duration

Improving Reliability and Availability

1. Fault avoidance: Prevent faults from occurring
  - No offee in server room
2. Fault tolerance: Prevent faults from becoming failures
  - Redundancy: ECC (error correcting code) for memory

3. Speed up repair: Only affects availability
    - Keep spare hard drive in drawer so it can be replaced quickly

Fault Tolerance Techniques

1. Checkpointing
    - Save state periodically
    - Detect errors -> restore state
    - Works well for many transient and intermittent faults
    - Can't take too long, this is treated as a service interruption
2. 2-way redundancy (detect)
    - Two modules do the same work and compare the results
    - Roll back if the results are different
3. 3-way redundancy (detect and recover)
    - Three modules (or more) do the same work and vote on the correct result
    - A fault can become an error in one module, but we can prevent a failure at the system level
    - Expensive, but can tolerate any fault in one module even if that module is intentionally designed to be malicious
        - Can't tolerate two modules failing

N Module Redundancy

1. Dual-module redundancy (N=2)
    - Detect but not correct one faulty module
2. Triple-module redundancy (N=3)
    - Detect and correct one faulty module
3. Five-module redundancy (N=5)
    - Space shuttle
    - 5 computers compute the result and vote
    - One wrong result in a vote -> normal operation
    - Two wrong results in a vote -> abort mission
        - No failure: 3 outvote the 2
        - Gives an opportunity to safely recover while three computers are still working
    - Three wrong results -> Can no longer promise things are correct

N Module Quiz

1. Have a computer and want to tolerate faults
2. Buy two more just like it and put on the same desk
3. Run every computation on all three computers, compare results, and take result where greater than or equal to two agree
4. Can we tolerate the following events?
    - Alpha particle strikes a processor (yes)
    - Building collapses (no)
    - Earthquake (no)
    - Mistake in processor design (no)
5. Replicating hardware with identical hardware in an identical location solves only the first case
    - To solve building/earthquake, need to geographically distribute
    - To solve mistake in design, need different architectures

Fault Tolerance for Memory and Storage

1. Dual-module and triple-module redundancy are overkill for these devices
2. Error detection and correction codes are commonly used
    - Store bits with extra information to detect and/or correct one or more bits of error

    - Parity: Add one extra bits (xor of all data bits)
        - Fault flips one bit -> parity does not match data
    - Error correction code (ECC): SECDED codes
        - SECDED: Single error correction, double error detection
        - Example: ECC DRAM modules
3. Disks use even fancier codes (Reed-Solomon)
    - Detect and correct multiple-bit errors (especially streaks of flipped bits)
        - This can occur when the head is misaligned for some period
    - RAID is another technique commonly employed

RAID

1. RAID: Redundant array of independent disks
    - Several disks playing the role of one disk
        - Can pretend to be larger, more reliable, or both larger and more reliable
    - Each of the disks is still detecting errors using codes
        - Know which disk has an error
2. Goals of RAID
    - Better performance
    - Normal read/write accomplishment even when there's a bad sector or an entire disk fails
        - These are problems a basic error detection/correction code can't fix
    - Not all RAID techniques address all of these problems
    - Numbered (RAID 0, RAID 1, . . .)

RAID 0

1. Uses a technique called striping to improve performance
    - While the head is positioned to read track 0, it can't read any other track
        - Must serialize all accesses
2. RAID 0 takes two disks and makes it look like one disk, which allows both to be read from simultaneously, providing 2x throughput
    - "Stripes" the data across the two disks
    - It's possible we can be unlucky and all the data is on the same disk
        - On average, we achieve 2x throughput
    - Less queuing delay because requests are handled faster
    - Reliability is worse

RAID 0 Reliability

1. Let f be the failure rate for a single disk
    - Failures/disk/second
2. For a single disk, MTTF = 1 / f
    - For disks, MTTF is also called MTTDL (mean time to data loss)
3. If we put N disks in RAID 0. . .
    - fn = N * f1
    - MTTFn = MTTDLn = MTTF1 / N
        - This means the MTTF for two disks is half of the MTTF for one disk

RAID 0 Quiz

1. RAID 0 array with four disks
    - One disk: 200GB, 10MB/s throughput, MTTF = 100000 hours
2. Our RAID 0 array has the following properties:
    - Can store 200 * 4 = 800 GB
    - Throughput = 10 * 4 = 40MB/s

- MTTF = 100000 / 4 = 25000 hours

RAID 1 Mirroring

1. RAID 1 stores the same data on both disks
    - Improves reliability
    - Write: Write to each disk
        - Same write performance as one disk alone (writes are simultaneous)
    - Read: Read any one disk
        - 2x throughput of one disk alone
    - Tolerate any faults that affect one disk (even entire disk failure)
        - ECC on each sector will detect errors, so we can simply use the other disk (can both detect and correct)

RAID 1 Reliability

1. Let f be the failure rate for a single disk
    - Failures/disk/second
2. For a single disk, MTTF = 1 / f
    - For disks, MTTF is also called MTTDL (mean time to data loss)
3. 2 disks in RAID 1
    - fn = N * f1
    - Both disks are okay until MTTF1 / 2
    - Remaining disk lives on for MTTF1
    - MTTDL(RAID1) = MTTF1/2 + MTTF1
        - Assumes no disk is replaced
    - However, by replacing the failed disk, we restore our RAID to working order

RAID 1 Reliability if Failed Disks are Replaced?

1. Both disks okay for MTTF1/2
2. Disk fails; have one working disk for MTTR
3. Both disks are okay again -> MTTF1/2
4. MTTDL(RAID1) = MTTF1/2 * (1 / MTTR / MTTF) = MTTF ˆ 2 / (2 * MTTR)
    - When MTTR << MTTF1, probability of second disk failing during MTTR can be approximated as MTTR / MTTF
    - MTTF is typically measured in years while MTTR is hours or days
    - Time to data loss is significantly higher than for a single disk
        - RAID 1 dramatically improves reliability

RAID 1 Quiz

1. RAID 1 array with two disks
    - One disk: 200GB, 10MB/s throughput, MTTF = 100000 hours
    - We replace the failed disk, MTTR = 24 hours
2. Our RAID 1 array has the following properties:
    - Can store 200 GB
    - Throughput (50% read, 50% write) = 20/3 + 10 * 2/3 = 40/3 = 13.33 MB/s
        - If the workload is 50/50 in terms of number of accesses, then it isn't also 50/50 in terms of time spent on reads and writes
    - MTTF = 100000 ˆ 2 / (2 * 24) = 208333333.33 hours

RAID 4

1. RAID 3 and 4 are very rarely used

- RAID 4 uses block-interleaved parity
- N disks
  - N-1 contain data, striped like RAID 0
  - 1 disk has parity blocks
- Parity bits are computed as the xor of each stripe on the other disks

2. If one disk fails, we can reconstruct the data on the first disk using the other disks combined with the parity bits
   - Only 1/N disks are spent on parity compared to RAID 1 where 1/2 are used
   - Considered a more general technique for mirroring
   - Write: Write 1 data disk and parity disk
   - Read: Read 1 disk

RAID 4 Performance and Reliability

1. Reads: Throughput of N-1 disks
2. Writes: 1/2 throughput of a single disk
   - Requires two accesses for every write
     - This is what RAID 5 addresses
3. MTTF
   - All disks are okay for MTTF/N
     - If we don't repair, MTTF = MTTF/N + MTTF/(N-1), which is worse than the MTTF for a single disk
     - If we repair, MTTF = MTTF/N * (MTTF/(N-1)/MTTR)
     - MTTF = MTTF ˆ 2 / (N * (N-1) * MTTR)

RAID 4 Write

1. To compute the new parity, we compare the new data against the old data, then XOR with the parity bit
   - Requires reading old data and old parity
   - Requires writing over previous data as well as parity
2. This means the parity disk is a bottleneck for writes
   - RAID 5 fixes the write performance

RAID 4 Quiz

1. RAID 4 array with five disks
   - One disk: 200GB, 10MB/s throughput, MTTF = 100000 hours
   - We replace the failed disk, MTTR = 24 hours
2. Our RAID 4 array has the following properties:
   - Can store 200 * (5-1) = 800 GB
   - Throughput (50% read, 50% write) = 40/9 + 5 * 8/9 = 80/9 = 8.89 MB/s
     - For reads, 10 * (5-1) = 40 MB/s
     - For writes, 10/2 = 5 MB/s
   - MTTF = 100000 ˆ 2 / (5 * (5-1) * 24) = 20833333 hours

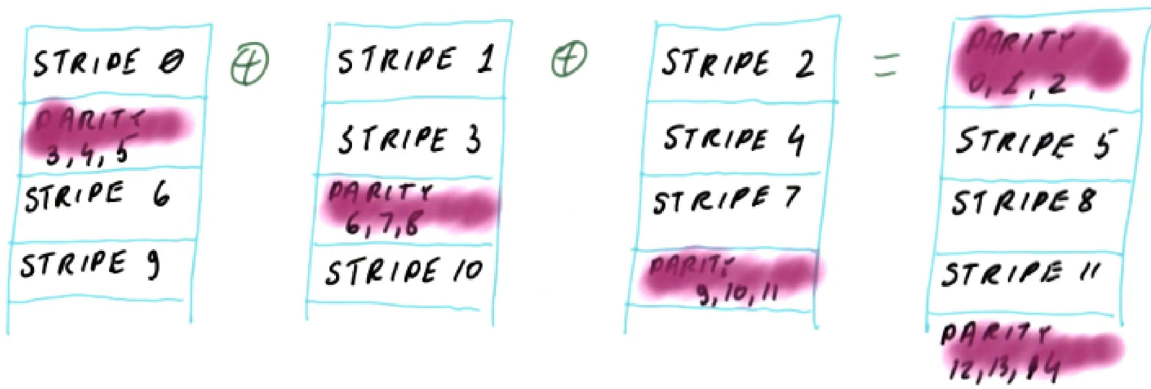
Parity Quiz

1. Use parity to detect bit-flips in our DRAM memory
2. Unprotected: Eight 1-bit 1024 x 1024 array
3. Want to add a parity bit for each 4 data bits
4. Should we:
   - Add two more 1-bit arrays for parity (yes)
   - Add 256 bits to every row in each array (no)

5. We should add two more 1-bit arrays so that if an entire array is lost, we can recover. This is not possible with the second option.
    - Similar to geographic distribution
    - This approach is also more scalable because we don't have to redesign our memory

RAID 5

1. RAID 5 uses distributed block-interleaved parity to protect the disks
    - Like RAID 4, but parity is spread among all disks
    - Read: N * throughput of 1 disk
    - Write: 4 accesses per write (2 for data block, 2 for parity)
        - These are distributed across disks -> N/4 * throughput of one disk
    - Reliability is same as RAID 4
        - Fails if more than one disk is lost



RAID 5 Distribution of Parity

RAID 5 Quiz

1. RAID 5 array with five disks
    - One disk: 200GB, 10MB/s throughput, MTTF = 100000 hours
    - We replace the failed disk, MTTR = 24 hours
2. Our RAID 5 array has the following properties:
    - Can store 200 * (5-1) = 800 GB
    - Throughput (50% read, 50% write) = 50/5 + 12.5 * 4/5 = 100/5 = 20 MB/s
        - For reads, 10 * 5 = 50 MB/s
        - For writes, 5 / 4 * 10 = 12.5 MB/s
    - MTTF = 100000 ^ 2 / (5 * (5-1) * 24) = 20833333 hours
3. RAID 5 improves throughput over RAID 4 without sacrificing storage or MTTF

RAID 6

1. Similar to RAID 5 but with two "parity" blocks per group
    - Can work when 2 failed stripes per group
    - One is a parity block
    - Second is a different type of check-block
    - When 1 disk fails, use parity
    - When 2 disks fail, solve equations to recover the content
2. Comparison of RAID 5 and RAID 6

- 2x overhead
- More write overhead (6 accesses for RAID 6 vs 4 for RAID 5)
- Disk fails, then another fails before we replace first
  - Only useful if there's a good chance that another disk will fail before we can replace the first failure; this is very low probability

RAID 6 is an Overkill

1. RAID 5: Disk fails, 3 days to replace
   - Very low probability of another failing in those three days
     - This assumes failures are independent
2. Failures can be related
   - RAID 5, 5 disks, 1 disk fails
   - System says "Replace disk 2"
   - Operator gets replacement disk, but accidentally pulls the wrong drive
     - Now we actually have two failed disks

Conclusion

1. Redundancy helps prevents faults from becoming failures
2. Next part of course focuses on multicore processors