Dependable Systems

Hardware Dependability - Redundancy

Dr. Peter Tröger

Sources:

Roland Trauner, IBM Mainframe Summit, Hasso Plattner Institute, 2012
IBM zEnterprise System Technical Guide, IBM RedBooks
Some images (C) Elena Dubrova, ESDLab, Kungl Tekniska Högskolan
Redundancy (Reiteration)

- Redundancy for **error detection** and **forward error recovery**

- Redundancy types: **spatial**, **temporal**, **informational** (presentation, version)
  - Redundant not mean identical functionality, just perform the same work

- **Static redundancy** implements error mitigation
  - Fault does not show up, since it is transparently removed
  - Examples: Voting, error-correcting codes, N-modular redundancy

- **Dynamic redundancy** implements error processing
  - After fault detection, the system is reconfigured to avoid a failure
  - Examples: Back-up sparing, duplex and share, pair and spare

- **Hybrid approaches**
System-Failure Response Strategies
[Sieworek / Swartz]

- System reliability
  - Nonredundant system
  - Redundant system
    - Fault detection
    - Fault tolerance
      - Masking redundancy
        - Reconfiguration
      - Dynamic redundancy
        - Retry
        - Online Repair
Redundancy

• Redundancy is never for free!
  • Hardware: Additional components, area, power, shielding, ...
  • Software: Development costs, maintenance costs, ...
  • Information: Extra hardware for decoding and encoding
  • Time: Faster processing (CPU) necessary to achieve same performance

• Tradeoff: Costs vs. benefit of redundancy; additional design and testing effort

• Sphere of replication [Mukherjee]
  • Identifies logical domain protected by the fault detection scheme
  • Questions: For which components are faults detected? Which outputs must be compared? Which inputs must be replicated?
Sphere of Replication

- Components outside the sphere must be protected by other means
- Level of output comparison decides upon fault coverage
- Larger sphere tends to decrease the required bandwidth on input and output
- More state changing happens just inside the sphere
- Vendor might be restricted on choice of sphere size
Masking / Static Redundancy: Voting

- **Exact voting**: Only one correct result possible
  - **Majority vote** for uneven module numbers
  - **Generalized median voting** - Select result that is the median, by iteratively removing extremes
  - **Formalized plurality voting** - Divide results in partitions, choose random member from the largest partition

- **Inexact voting**: Comparison at high level might lead to multiple correct results
  - **Non-adaptive voting** - Use allowable result discrepancy, put boundary on discrepancy minimum or maximum (e.g. 1,4 = 1,3)
  - **Adaptive voting** - Rank results based on past experience with module results
    - Compute the correct value based on „trust“ in modules from experience
    - Example: Weighted sum $R=W_1R_1 + W_2R_2 + W_3R_3$ with $W_1+W_2+W_3=1$
Static Redundancy: N-Modular Redundancy

- Fault is transparently removed on detection
- **Triple-modular redundancy (TMR)**
  - 2/3 of the modules must deliver correct results
- **Generalization with N-modular redundancy (NMR)**
  - \( m+1/N \) of the modules must deliver correct result, with \( N=2m+1 \)
- Standard case without any redundancy is called **simplex**

\[
R_{TMR} = R_V \cdot R_{2-of-3} = R_V (R_M^3 + 3R_M^2(1 - R_M))
\]
N-Modular Redundancy (with perfect voter)

\[ R_{NMR} = \sum_{i=0}^{m} \binom{N}{i} (1 - R)^i R^{N-i} \]

\[ \binom{n}{k} = \frac{n!}{k!(n-k)!} \]

\[ R_{2-of-3} = \binom{3}{3} (1 - R)^0 R^3 + \binom{3}{1} (1 - R) R^2 \]

\[ R_{2-of-3} = R^3 + 3(1 - R) R^2 \]

\[ R_{3-of-5} = \ldots \]
TMR Reliability

• TMR is appropriate if $R_{TMR} > R_M$ (for given $t$)

• TMR with perfect voter only improves system reliability when $R_M > 0.5$

• Voter needs to have $R_V > 0.9$ to reach $R_{TMR} > R_M$
Imperfect Voters

- Redundant voters
  - Module errors do not propagate
  - Voter errors propagate only by one stage
- Assumption of multi-step process, final voter still needed
Hardware Voting

- Smallest hardware solution is the 1-bit majority voter
  - \( f = ab + ac + bc \)
  - Delivers the bit that has the majority
  - Requires 2 gate delays and 4 gates
- Hardware voting can become expensive
  - 128 gates and 256 flip-flops for 32-bit voter
- Input must be synchronized
  - Central clock source may be single point of failure
  - Can be solved by special event latching
Dynamic Redundancy

- Reconfiguration of the system in response to an error state
  - Prevents error propagation
  - Triggered by internal fault detection in the unit, or external error detection based on the outputs
- Dynamic redundancy combines error confinement with fault detection
  - Still questions of coverage and diagnosability
- On transient errors, good modules may be deactivated
  - Typically solved by combination of dynamic redundancy with retry approach
- Typical approaches: Duplex, sparing, degradation, compensation
Duplex Systems

- Reconfigurable duplication: Have relevant modules redundant, switch on failure
- Identification on mismatch ("test")
  - Self-diagnostics procedure
  - Self-checking logic
  - Watchdog timer, e.g. for having components resetting each other (e.g. split brain)
  - Outside arbiter for signatures or black box tests
- Test interval depends on application scenario - each clock period / bus cycle / ...
- Also called **dual-modular redundancy**
- Reliability computation as with parallel / serial component diagram
Back-Up Sparing

• Combination of working module and a set of spare modules (‘replacement parts’)

• **Hot spares**: Receive input with main modules, have results immediately

• **Warm spares**: Are running, but receive input only after switching

• **Cold spares**: Need to be started before switching
Pair and Spare

- Special cases for combination of duplex (with comparator) and sparing (with switch)

- **Pair and spare** - Multiple duplex pairs, connected as standby sparing setup
  
  - Two replicated modules operate as duplex pair (lockstep execution), connected by comparator as voting circuit
  
  - Same setting again as spare unit, spare units connected by switch
  
  - On module output mismatch, comparators signal switch to perform failover
  
- Commercially used, e.g. Stratus XA/R Series 300
Graceful Degradation

- Performance design of the system allows continued operation with spares
  - Many commercial systems supports this, but lack automated error processing
  - Example: Operating system support for CPU off-lining, but no MCA handling

- Designed-In Resources:
  - Replaceable or bypass-able components (f.e. caches, disks, processors)
  - Support for operation with degraded performance

- Added-On Resources:
  - Redundant units used for excess capacity during normal operation
  - Still non-degraded performance on failure

- Interconnect reconfiguration: Use alternative paths in the network
  - Hardware solutions in telco industry, today replaced by software solutions
Example: Spanning Tree Protocol

- Modern implementation of interconnect reconfiguration for dynamic redundancy
- Bridges for connecting different Ethernet sub-networks
- By default no coordination, better products use the spanning tree protocol
  - Explicit removal of redundant paths (loops), while still supporting all point-to-point communication
  - Each bridge has its own MAC address, protocol based on broadcast
  - Create a tree of bridges, starting from a chosen root bridge
  - All paths start from the root bridge
  - Ports participating in redundant paths have to be switched off
  - Cost model for paths to make a choice (root distance, speed)
Example: Spanning Tree Protocol

• Determine root bridge
  • Send your ID (MAC address) to a multicast group, smallest ID wins

• Each non-root bridge determines the 'cheapest' path to the root bridge
  • This port becomes the root port (RP)

• For multiple bridges in a segment, the 'cheapest' representative is elected - designated port (DP)

• All ports that are not DP or RP are deactivated - blocked port (BP)
Hybrid Approaches

- **N-modular redundancy with spares**
  - Also called *hybrid redundancy*
  - System has basic NMR configuration
  - Disagreement detector replaces modules with spares if their output is not matching the voting result
  - Reliability as long as the spare pool is not exhausted
  - Improves fault masking capability of NMR
    - Can **tolerate two faults with one spare**, while classic NMR would need 5 modules with majority voting to tolerate two faults

'Thumbnail Diagram Description: Diagram illustrating a hybrid redundancy system with a switch, voter, and disagreement detector.'
TMR with Spares

- Basic reliability computation based on the assumption of similar module failure rates in spares and non-spares

- At least any two of all S+3 modules must survive

Comparison TMR vs. TMR/S vs. NMR

#Units = 2N + 1
Hybrid Approaches

- **Self-purging redundancy**
  - Active redundant modules, each can remove itself from the system if faulty
  - Basic idea: Test for agreement with the voting result, otherwise 0

- If module output does not match to system output, 0 is delivered
- Works fine with threshold voters
Hybrid Approaches

- **Sift-out modular redundancy** (N-2), no voter required
  - Pair-wise comparison of module outputs by **comparator**
    - N inputs and N-over-2 outputs
  - **Detector** uses these signals to identify the faulty module, includes also memory cells for failed modules
  - **Collector** sifts out the faulty input, based on information from detector
Hybrid Approaches

• Triple Duplex Architecture

  • TMR with duplex modules, used in the Shinkansen (Japanese train)
  • Fault masking with comparator, no more contribution to voting from faulty one
  • Allows tolerating another fault in the further operation, since comparator localizes again the faulty module
  • Adds again fault location capability to redundancy scheme
  • Supports also hot plugging of deactivated components
The Real World of Hardware Redundancy - Replacement Frequencies [Schroeder 2007]

<table>
<thead>
<tr>
<th>Component</th>
<th>HPC1 Component</th>
<th>%</th>
<th>Component</th>
<th>COM1 Component</th>
<th>%</th>
<th>Component</th>
<th>COM2 Component</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hard drive</td>
<td></td>
<td>30.6</td>
<td>Power supply</td>
<td></td>
<td>34.8</td>
<td>Hard drive</td>
<td></td>
<td>49.1</td>
</tr>
<tr>
<td>Memory</td>
<td></td>
<td>28.5</td>
<td>Memory</td>
<td></td>
<td>20.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Misc/Unk</td>
<td></td>
<td>14.4</td>
<td>Case</td>
<td></td>
<td>11.4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td></td>
<td>12.4</td>
<td>Fan</td>
<td></td>
<td>8.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCI motherboard</td>
<td></td>
<td>4.9</td>
<td>CPU</td>
<td></td>
<td>2.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Controller</td>
<td></td>
<td>2.9</td>
<td>RAID card</td>
<td></td>
<td>4.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QSW</td>
<td></td>
<td>1.7</td>
<td>Memory</td>
<td></td>
<td>3.4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power supply</td>
<td></td>
<td>1.6</td>
<td>NIC Card</td>
<td></td>
<td>1.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MLB</td>
<td></td>
<td>1.0</td>
<td>LV Power Board</td>
<td></td>
<td>0.6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCSI BP</td>
<td></td>
<td>0.3</td>
<td>CPU heatsink</td>
<td></td>
<td>0.6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 760 node cluster, 2300 disks
- ISP, multiple sites, 26700 disks
- ISP, multiple sites, 9200 machines, 39000 disks
## Sources of Outages

### Pre z9 - Hrs/Year/Syst -

<table>
<thead>
<tr>
<th></th>
<th>Prior Servers</th>
<th>z9 EC</th>
<th>Z10 EC</th>
<th>z196</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unscheduled Outages</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Scheduled Outages</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Planned Outages</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Preplanning requirements</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Power &amp; Thermal Management</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

**Temperature = Silicon Reliability Worst Enemy**

**Wearout = Mechanical Components Reliability Worst Enemy.**
IBM System z

- Machine-Check-Handling mechanism in z/Series
  - Equipment malfunction detection
  - Permit automatic recovery
  - Error states are reported by machine-check interruption
- Data error detection through information redundancy
- Recovery from machine-detected error states
  - Error checking and correction - use **circuitry redundancy**
  - CPU retry - **checkpoint** written at instruction-based synchronization points
  - Channel-subsystem recovery - **restart** of I/O components
  - Unit deletion - automated **degradation** of malfunctioning units
IBM System z - Processor Books

z196 Cache / Book Topology

Fully connected 4 Book system:

- 96 total cores
- Total system cache
  - 768 MB shared L4 (eDRAM)
  - 576 MB L3 (eDRAM)
  - 144 MB L2 private (SRAM)
  - 19.5 MB L1 private (SRAM)

- 4 PU’s (cores) per CP + co-processors
- SC: Storage Control
  - 96-192 MB L4 cache per SC, accessible from other MCMs
- FBC: Fabric book connectivity
- Support for dynamic book addition and repair
- 2+1 redundancy for book power supply

Multi-Chip Module (MCM)
IBM System z

z196 RAS Design of
Full Redundant I/O Subsystem – of existing IO cage and drawers

### Fully Redundant I/O Design
- SAP / CP sparing
- SAP Reassignment
- I/O Reset & Failover
- I/O Mux Reset / Failover
- Redundant I/O Adapter
- Redundant I/O interconnect
- Redundant Network Adapters
- Redundant ISC links
- Redundant Crypto processors
- I/O Switched Fabric
- Network Switched/Router Fabric
- High Availability Plugging Rules
- I/O and coupling fanout rebalancing on CBA
- Channel Initiated Retry
- High Data Integrity Infrastructure
- I/O Alternate Path
- Network Alternate Path
- Virtualization Technology
IBM System z196

- Prepared for Unscheduled Outages
  - Advanced Memory RAIM (Redundant Array of Independent Memory) design
  - Enhanced Reed-Solomon code (ECC) – 90B/64B
  - Protection against Channel/DIMM failures
  - Chip marking for fast DRAM replacements
  - Mirrored Key cache
  - Improved chip packaging
  - Improved condensation management
  - Integrated TCP/IP checksum generation/checking
  - Integrated EPO switch cover (protecting the switch during repair actions)
  - Continued focus on Firmware

- Prepared for Scheduled Outages
  - Double memory data bus lane sparing (reducing repair actions)
  - Single memory clock bus sparing
  - Field Repair of interface between processor chip and cache chip and between cache chips (fabric bus)
  - Fast bitline delete on L3/L4 cache (largest caches)
  - Power distribution using N+2 Voltage Transformation Modules (VTM)
  - Blower management by sensing altitude and humidity
    - Redundant (N+2) humidity sensors
    - Redundant (N+2) altitude sensors
  - Unified Resource Manager for zBX
Memory Redundancy

• Redundancy of memory data for error masking
• Replication / coding at different levels
• Examples
  • STAR (Self-testing and self-repairing computer, for early spacecrafts), 1971
  • COMTRAC (Computer-aided traffic control system for Shinkansen train system)
  • Stratus (Commercial fault-tolerant system)
  • 3B20 by AT & T (Commercial fault-tolerant system)
  • Most modern memory controllers in servers
Coding Checks in Memory Hardware

• **Primary memory**
  
  • Parity code
  
  • m-out-of-n resp. m-of-n resp. m/n code
  
  • Checksumming
  
  • Berger Code
  
  • Hamming code

• **Secondary storage**
  
  • RAID codes
  
  • Reed-Solomon code
Hamming Distance (Richard Hamming, 1950)

- **Hamming distance**: Number of positions on which two words differ
  - Alternative definition: Number of necessary substitutions to make A to B
  - Alternative definition: Number of errors that transform A to B

- **Minimum Hamming distance**: Minimum distance between any two code words
  - To *detect* $d$ single-bit errors, a code must have a min. distance of at least $d + 1$
    - If at least $d+1$ bits change in the transmitted code, a new (valid) code appears
  - To *correct* $d$ single-bit errors, the minimum distance must be $2d+1
Parity Codes

• Add parity bit to the information word
  • Even parity: If odd number of ones, add one to make the number of ones even
  • Odd parity: If even number of ones, add one to make the number of ones odd

• Variants
  • Bit-per-word parity
  • Bit-per-byte parity
    (Example: Pentium data cache)
  • Bit-per-chip parity
  • ...
Two-Dimensional Parity

Example: Odd Parity

<table>
<thead>
<tr>
<th>Column Parity Register</th>
<th>Row Parity Register</th>
<th>Parity Error?</th>
<th>Overall parity check bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 0 1 0 1 0</td>
<td>0</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>1 0 0 0 0 1</td>
<td>1</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>1 1 0 1 0</td>
<td>1</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>0 1 1 1 0 1</td>
<td>0</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

n bits/word:

- 0 0 1 1 1 1
- 1 0 0 0 0 1
- 1 1 0 1 0
- 0 1 1 1 0 1

k words:

- 0 0 1 1 1 1
- 1 0 0 0 0 1
- 1 1 0 1 0
- 0 1 1 1 0 1

Allows fault location.
# Code Properties

<table>
<thead>
<tr>
<th>Code</th>
<th>Bits / Word</th>
<th>Number of possible words</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Even Parity</td>
<td>4</td>
<td>8</td>
<td>Any single bit error, no double-bit error, not all-0s or all 1s errors</td>
</tr>
<tr>
<td>Odd Parity</td>
<td>4</td>
<td>8</td>
<td>Any single bit error, no double-bit error, all-0s or all 1s errors</td>
</tr>
<tr>
<td>2/4</td>
<td>4</td>
<td>6</td>
<td>Any single bit error, 33% of double bit errors</td>
</tr>
</tbody>
</table>
Checksumming

- General idea: Multiply components of a data word and add them up to a checksum
  - Good for large blocks of data, low hardware overhead
  - Might require substantial time, memory overhead
  - 100% coverage for single faults

- Example: ISBN 10 check digit
  - Final number of 10 digit ISBN number is a check digit, computed by:
    - Multiply all data digits by their position number, starting from right
    - Take sum of the products, and set the check digit so that result MOD 11 = 0
    - Check digit „01“ is represented by X

- More complex approaches get better coverage (e.g. CRC) for more CPU time
Hamming Code

- Every data word with n bits is extended by k control bits

- **Control bits** through standard parity approach (even parity), implementable with XOR
  - Inserted at **power-of-two positions**
  - Parity bit $p_x$ is responsible for all position numbers with the **X-least significant bit** set (e.g. $p_1$ responsible for 'red dot' positions)

- Parity bits check overlapping parts of the data bits, computation and check are cheap, but significant overhead in data

- Minimum Hamming distance of three (single bit correction)

- Application in DRAM memory
Hamming Code for 26 bit word size
Hamming Code

- Hamming codes are known to produce 10% to 40% of data overhead
- Syndrome determines which bit (if any) is corrupted
- Example ECC
  - Majority of "one-off" soft errors in DRAM happens from background radiation
  - Denser packages, lower voltage for higher frequencies
  - Most common approach is the SECDEC Hamming code
    - "Single error correction, double error detection" (SECDEC)
    - Hamming code with additional parity
  - Modern BIOS implementations perform threshold-based warnings on frequent correctable errors

(C) Z. Kalbarczyk
Memory Redundancy

- Standard technology in DRAMs
  - Bit-per-byte **parity**, check on read access
  - Implemented by additional parity memory chip
  - **ECC** with Hamming codes - 7 check bits for 32 bit data words, 8 bit for 64 bit
    - Leads to 72 bit data bus between DIMM and chipset
    - Computed by memory controller on write, checked on read
    - Study by IBM: ECC memory achieves R=0.91 over three years
    - Can correct single bit errors and detect double bit errors
- Hewlett Packard **Advanced ECC** (1996)
  - Can detect and correct single bit and double bit errors
Memory Redundancy

- IBM ChipKill
  - Originally developed for NASA Pathfinder project, now in X-Series
  - Corrects up to 4 bit errors, detects up to 8 bit errors
  - Implemented in chipset and firmware, works with standard ECC modules
  - Based on striping approach with parity checks (similar to RAID)
  - 72 bit data word is split in 18 bit chunks, distributed on 4 DIMM modules
  - 18 DRAM chips per module, one bit per chip

- HP Hot Plug RAID Memory
  - Five memory banks, fifth bank for parity information
  - Corrects single bit, double bit, 4-bit, 8-bit errors; hot plugging support
Memory Redundancy

- Dell PowerEdge Servers, 2005 (taken from www.dell.com)
Memory Redundancy

- Fujitsu System Board D2786 for RX200 S5 (2010)
- Independent Channel Mode: Standard operational module, always use first slot
- Mirrored Channel Mode: Identical modules on slot A/B (CPU1) and D/E (CPU2)
IBM System z - Memory RAID

• System z10 EC memory design
  
  • Four Memory Controllers (MCUs) organized in two pairs, each MCU with four redundant channels
  
  • 16 to 48 DIMMs per book, plugged in groups of 8
  
  • 8 DIMMs (4 or 8 GB) per feature, 32 or 64 GB physical memory per feature
  
  • 64 to 384 GB physical memory per book = 64 to 384 GB for use (HSA and customer)

• z196 memory design:
  
  • Three MCUs, each with five channels. The fifth channel in each z196 MCU is required to implement Redundant Array of Independent Memory (RAIM)
    
    • Detected and corrected: Bit, lane, DRAM, DIMM, socket, and complete memory channel failures, including many types of multiple failures
IBM System z196 - Memory RAID

Layers of Memory Recovery

ECC
- Powerful 90B/64B Reed Solomon code

DRAM Failure
- Marking technology; no half sparing needed
- 2 DRAM can be marked
- Call for replacement on third DRAM

Lane Failure
- CRC with Retry
- Data – lane sparing
- CLK – RAIM with lane sparing

DIMM Failure (discrete components, VTT Reg.)
- CRC with Retry
- Data – lane sparing
- CLK – RAIM with lane sparing

DIMM Controller ASIC Failure
- RAIM Recovery

Channel Failure
- RAIM Recovery
IBM z10 EC Memory Structure

- Level 2 Cache
- Key Cache
- MCU 0
- MCU 1
- MCU 2
- MCU 3

DATA
CHECK
ECC

Extra column provides RAIM function
IBM System z196 - RAIM

Level 3 Cache

MCU 0
Key Cache
16B
2B

MCU 1
Key Cache
16B
2B

MCU 2
Key Cache
16B
2B

DATA
CHECK
ECC
RAIM Parity

Extra column provides RAIM function
Disk Redundancy

• Typical measure is the annual failure rate \((AFR)\) - average number of failures / year

\[
AFR = \frac{1}{MTBF_{years}} = \frac{8760}{MTBF_{hours}}
\]

• Can be interpreted as failure probability during a year

• MTBF = Mean time **before** failure, here

• Disk MTTF: On average, one failure takes place in the given disk hours

• Example: Seagate Barracuda ST3500320AS: MTTF=750000h=85.6 years
  • With thousand disks, on average every 750h (a month) some disk fails
  • Measured by the manufacturer under heavy load and physical stress

• \(AFR=0.012\)
RAID

- **Redundant Array of Independent Disks (RAID)** [Patterson et al. 1988]
  - Improve I/O performance and/or reliability by building *raid groups*
  - Replication for information reconstruction on disk failure (*degrading*)
  - Requires computational effort (dedicated controller vs. software)
  - Assumes failure independence
RAID Reliability Comparison

• Treat disk failing as Bernoulli experiment - independent events, identical probability

• Probability for k events of probability p in n runs

\[ B_{n,p}(k) = p^k (1 - p)^{n-k} \binom{n}{k} \]

• Probability for a failure of a RAID 1 mirror - all disks unavailable:

\[ p_{all\ fail} = \binom{n}{n} p_{fail}^n (1 - p_{fail})^0 = p_{fail}^n \]

• Probability for a failure of a RAID 0 strip set - any faults disk leads to failure:

\[ p_{any\ fail} = 1 - p_{all\ work} \]
\[ = 1 - \binom{n}{n} (1 - p_{fail})^n p_{fail}^0 \]
\[ = 1 - (1 - p_{fail})^n \]
RAID MTTF Calculation [Patterson]

- Works for RAID levels were second outage during repair is fatal
- Basic idea is that groups of data disks are protected by additional check disks
  - D - Total number of data disks
  - G - Number of data disks in a group (e.g. G=1 in RAID1)
  - C - Number of redundant check disks (parity / mirror) in a group (e.g. C=1 in RAID1)
  - \(n_G = \frac{D}{G} = \text{number of groups}, \ G+C: \text{Number of disks in a group}

\[
MTTF_{Group} = \frac{MTTF_{Disk}}{G + C} \cdot \frac{1}{p_{SecondFailureDuringRepair}}
\]
RAID MTTF Calculation [Patterson]

- Assuming exponential distribution, the probability for a second disk failure during the repair time can be determined by:

\[
P_{\text{Second Failure}} = \frac{MTTR}{MTTF_{\text{Disk}}} \frac{1}{G+C-1}
\]

- So:

\[
MTTF_{\text{Group}} = MTTF_{\text{Disk}} \frac{1}{G+C} \frac{1}{P_{\text{Second Failure During Repair}}}
\]

\[
MTTF_{\text{Raid}} = \frac{MTTF_{\text{Group}}}{n_G}
\]

\[
= \frac{MTTF_{\text{Disk}}^2}{(G+C) \times n_G \times (G+C-1) \times MTTR}
\]
RAID 0

- **Raid 0** - Block-level striping
  - I/O performance improvement with many channels and drives
    - One controller per drive
  - Optimal stripe size depends on I/O request size, random vs. sequential I/O, concurrent vs. single-threaded I/O
    - Fine-grained striping: Good load balancing, catastrophic data loss
    - Coarse-grained striping: Good recovery for small files, worse performance
    - One option: Strip size = Single-threaded I/O size / number of disks
  - Parallel read supported, but positioning overhead for small concurrent accesses
  - No fault tolerance

\[
MTTF_{Raid0} = \frac{MTTF_{Disk}}{N}
\]
RAID 1

• **Raid 1** - Mirroring and duplexing
  
  • Duplicated I/O requests
  
  • Decreasing write performance, up to double read rate of single disk
    
    • RAID controller might allow concurrent read and write per mirrored pair
  
  • Highest overhead of all solutions, smallest disk determines resulting size
  
  • Reliability is given by probability that one disk fails and the second fails while the first is repaired

  • With $D=1$, $G=1$, $C=1$ and the generic formula, we get

\[
MTTF_{Raid1} = \frac{MTTF_{Disk}}{2} \cdot \frac{MTTF_{Disk}}{MTTR_{Disk}}
\]

(C) Wikipedia
Raid 2/3

- **Raid 2** - Byte-level striping with Hamming code-based check disk
  - No commercial implementation due to ECC storage overhead
  - Online verification and correction during read
- **Raid 3** - Byte-level striping with dedicated XOR parity disk
  - All data disks used equally, one XOR parity disk as bottleneck (C=1)
  - Bad for concurrent small accesses, good sequential performance (streaming)
  - Separate code is needed to identify a faulty disk
  - Disk failure has only small impact on throughput
  - RAID failure if more than one disk fails:
    \[
    MTTF_{Raid3} = \frac{MTTF_{Disk}}{D + C} \times \frac{MTTF_{Disk}}{MTTR_{Disk}}
    \]
# Parity With XOR

- **Self-inverse operation**

- \(101 \text{ XOR } 011 = 110, \text{ } 110 \text{ XOR } 011 = 101\)

<table>
<thead>
<tr>
<th>Disk</th>
<th>Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 1 0 0 1 0 0 1</td>
</tr>
<tr>
<td>2</td>
<td>0 1 1 0 1 1 1 0</td>
</tr>
<tr>
<td>3</td>
<td>0 0 0 1 0 0 1 1</td>
</tr>
<tr>
<td>4</td>
<td>1 1 1 0 1 0 1 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Disk</th>
<th>Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 1 0 0 1 1 1 1</td>
</tr>
<tr>
<td>Parity</td>
<td>0 1 0 1 1 1 1 1</td>
</tr>
<tr>
<td>3</td>
<td>0 0 0 1 0 0 1 1</td>
</tr>
<tr>
<td>4</td>
<td>1 1 1 0 1 0 1 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Disk</th>
<th>Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 1 0 0 1 0 0 1</td>
</tr>
<tr>
<td>Parity</td>
<td>0 1 0 1 1 1 1 1</td>
</tr>
<tr>
<td>3</td>
<td>0 0 0 1 0 0 1 1</td>
</tr>
<tr>
<td>4</td>
<td>1 1 1 0 1 0 1 1</td>
</tr>
</tbody>
</table>

**Hot Spare**

<table>
<thead>
<tr>
<th>Disk</th>
<th>Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 1 0 0 1 0 0 1</td>
</tr>
<tr>
<td>Parity</td>
<td>0 1 0 1 1 1 1 1</td>
</tr>
<tr>
<td>3</td>
<td>0 0 0 1 0 0 1 1</td>
</tr>
<tr>
<td>4</td>
<td>1 1 1 0 1 0 1 1</td>
</tr>
<tr>
<td>Hot Spare</td>
<td>0 1 1 0 1 1 1 0</td>
</tr>
</tbody>
</table>
RAID 4 / 5

- Raid 4 - Block-level striping with dedicated parity disk
- RAID 3 vs. RAID 4: Allows concurrent block access
- Raid 5 - Block-level striping with distributed parity
- Balanced load as with Raid 0, better reliability
- Bad performance for small block writing
- Most complex controller design, difficult rebuild
- When block in a stripe is changed, old block and parity must be read to compute new parity
  - For every changed data bit, flip parity bit

\[
MTTF_{Raid5} = \frac{MTTF_{Disk}}{N} \cdot \frac{MTTF_{Disk}}{N-1} \cdot \frac{MTTR_{Disk}}{MTTTR_{Disk}}
\]

(C) Wikipedia
RAID 6 / 01 / 10

- Raid 6 - Block-level striping with two parity schemes
  - Extension of RAID5, can sustain multiple drive failures at the same time
  - High controller overhead to compute parities, poor write performance

- Raid 01 - Every mirror is a Raid 0 stripe (min. 4 disks)

- Raid 10 - Every stripe is a Raid 1 mirror (min. 4 disks)

- RAID DP - RAID 4 with second parity disk
  - Additional parity includes first parity + all but one of the data blocks (diagonal)
  - Can deal with two disk outages
RAID Analysis (Schmidt)

• Take the same number of disks in different constellations
  • $AFR_{Disk} = 0.029$, MTTR=8h

• RAID5 has bad reliability, but offers most effective capacity

• In comparison to RAID5, RAID10 can deal with two disk errors

• Also needs to consider different resynchronization times
  • RAID10: Only one disk needs to be copied to the spare
  • RAID5 / RAID-DP: All disks must be read to compute parity

• Use RAID01 only in 2+2 combination
## RAID Analysis (TecChannel.de)

<table>
<thead>
<tr>
<th></th>
<th>RAID 0</th>
<th>RAID 1</th>
<th>RAID 10</th>
<th>RAID 3</th>
<th>RAID 4</th>
<th>RAID 5</th>
<th>RAID 6</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of drives</strong></td>
<td>n &gt; 1</td>
<td>n = 2</td>
<td>n &gt; 3</td>
<td>n &gt; 2</td>
<td>n &gt; 2</td>
<td>n &gt; 2</td>
<td>n &gt; 3</td>
</tr>
<tr>
<td><strong>Capacity overhead (%)</strong></td>
<td>0</td>
<td>50</td>
<td>50</td>
<td>100 / n</td>
<td>100 / n</td>
<td>100 / n</td>
<td>200 / n</td>
</tr>
<tr>
<td><strong>Parallel reads</strong></td>
<td>n</td>
<td>2</td>
<td>n / 2</td>
<td>n - 1</td>
<td>n - 1</td>
<td>n - 1</td>
<td>n - 2</td>
</tr>
<tr>
<td><strong>Parallel writes</strong></td>
<td>n</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>n / 2</td>
<td>n / 3</td>
</tr>
<tr>
<td><strong>Maximum read throughput</strong></td>
<td>n</td>
<td>2</td>
<td>n / 2</td>
<td>n - 1</td>
<td>n - 1</td>
<td>n - 1</td>
<td>n - 2</td>
</tr>
<tr>
<td><strong>Maximum write throughput</strong></td>
<td>n</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>n / 2</td>
<td>n / 3</td>
</tr>
</tbody>
</table>
Software RAID

- Software layer above block-based device driver(s)
- Windows Desktop / Server, Mac OS X, Linux, ...
- Multiple problems
  - Computational overhead for RAID levels beside 0 and 1
  - Boot process
  - Legacy partition formats
- Driver-based RAID
  - Standard disk controller with special firmware
  - Controller covers boot stage, device driver takes over in protected mode
Disk Redundancy: Google

- Failure Trends in a Large Disk Drive Population [Pinheiro2007]
  - > 100,000 disks, SATA / PATA consumer hard disk drives, 5400 to 7200 rpm
  - 9 months of data gathering in Google data centers
  - Statistical analysis of SMART data

- Failure event: „A drive is considered to have failed if it was replaced as part of a repairs procedure.“

- Prediction models based on SMART only work in 56% of the cases
Disk Redundancy: Google

- Failure rates are correlated with drive model, manufacturer and drive age
- Indication for infant mortality
- Impact from utilization (25th percentile, 50-75th percentile, 75th percentile)
  - Reversing effect in third year - "Survival of the fittest" theory
Disk Redundancy: Google

- Temperature effects only at high end of temperature range, with old drives
IBM System z - Redundant I/O

• Each processor book has up to 8 dual port fanouts

• Direct data transfer between memory and PCI/e (8 GBps) or Infiniband (6 GBps)

• Optical and copper connectivity supported

• Fanout cards are hot-pluggable, without loosing the I/O connectivity

• Air-moving devices (AMD) have N+1 redundancy for fanouts, memory and power
IBM System z - Redundant I/O

- PCI/e I/O drawer supports up to 32 I/O cards from fanouts in 4 domains
- One PCI/e switch card per domain
- Two cards provide backup path for each other (e.g. with cable failure)
- 16 cards max. per switch

Figure 4-2 illustrates the I/O structure of a z196. An InfiniBand (IFB) cable connects the HCA2-C fanout to an IFB-MP card in the I/O cage. The passive connection between two IFB-MP cards allows for redundant I/O interconnection. The IFB cable between an HCA2-C fanout in a book and each IFB-MP card in the I/O cage supports a 6 GBps bandwidth.
IBM System z - Redundant I/O

The InfiniBand and PCIe fanouts are located in the front of each book. Each book has eight fanout slots. They are named D1 to DA, top to bottom; slots D3 and D4 are not used for fanouts. Six types of fanout cards are supported by z196. Each slot holds one of the following six fanouts:

1. **/SM590000 Host Channel Adapter (HCA2-C):** This copper fanout provides connectivity to the IFB-MP card in the I/O cage and I/O drawer.
2. **/SM590000 PCIe Fanout:** This copper fanout provides connectivity to the PCIe switch card in the PCIe I/O drawer.
3. **/SM590000 Host Channel Adapter (HCA2-O (12xIFB)):** This optical fanout provides 12x InfiniBand coupling link connectivity up to 150 meters distance to a z196, z114, System z10 and System z9.
4. **/SM590000 Host Channel Adapter (HCA2-O LR (1xIFB)):** This optical long range fanout provides 1x InfiniBand coupling link connectivity up to 10 km unrepeated distance to a z196, z114 and System z10 servers.
5. **/SM590000 Host Channel Adapter (HCA3-O (12xIFB)):** This optical fanout provides 12x InfiniBand coupling link connectivity up to 150 meters distance to a z196, z114 and System z10, cannot communicate with an HCA1-O fanout on z9.
6. **/SM590000 Host Channel Adapter (HCA3-O LR (1xIFB)):** This optical long range fanout provides 1x InfiniBand coupling link connectivity up to 10 km unrepeated distance to a z196, z114 and System z10 servers.

The HCA3-O LR (1xIFB) fanout comes with 4 ports and each other fanout comes with two ports.

Figure 4-10 illustrates the IFB connection from the CPC cage to an I/O cage and an I/O drawer, and the PCIe connection from the CPC cage to an PCIe I/O drawer.