SAND2012-9540P

RAID: Motivation and Implementation

Matthew L. Curry, Ph.D.
Senior Member of Technical Staff

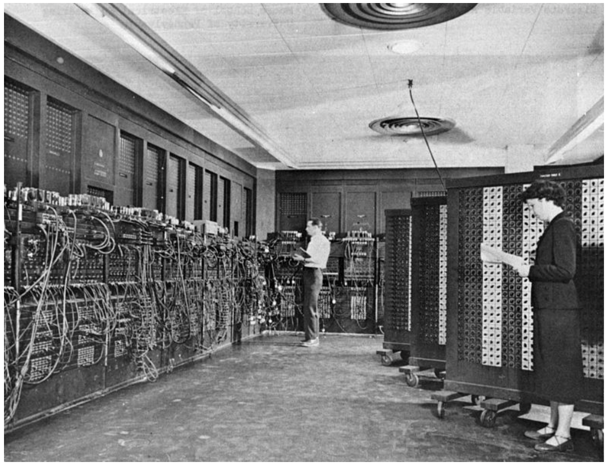
Scalable System Software
Sandia National Laboratories
Albuquerque, NM, USA
mlcurry@sandia.gov

Exceptional service in the national interest

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Computers and I/O (Input/Output)

- Computers have long been attractive for their ability to perform calculations quickly
  - From the ENIAC (5,000 ops/s in 1946) to Sequoia (20,000,000,000,000,000 ops/s in 2012)
- The largest computers spend of most of their time generating new data from old data
  - Often defensively
- I/O subsystems are used to guide program execution through the maintenance of state



I/O is a Bottleneck

- Computer programs alternate between two phases of operation
  - Compute phase – Generating results
  - I/O phase – Storing data from last compute phase and/or retrieving data for next
- Often, these cannot overlap!
- CPU processes improve performance, disks are much more tied to the limits of materials physics
- Need to find a way to make data storage and retrieval scale with the ever increasing compute power it services
  - Parallelism

RAID

- **R**edundant
- **A**rray of
- **I**nexpensive
- **D**isks

RAID

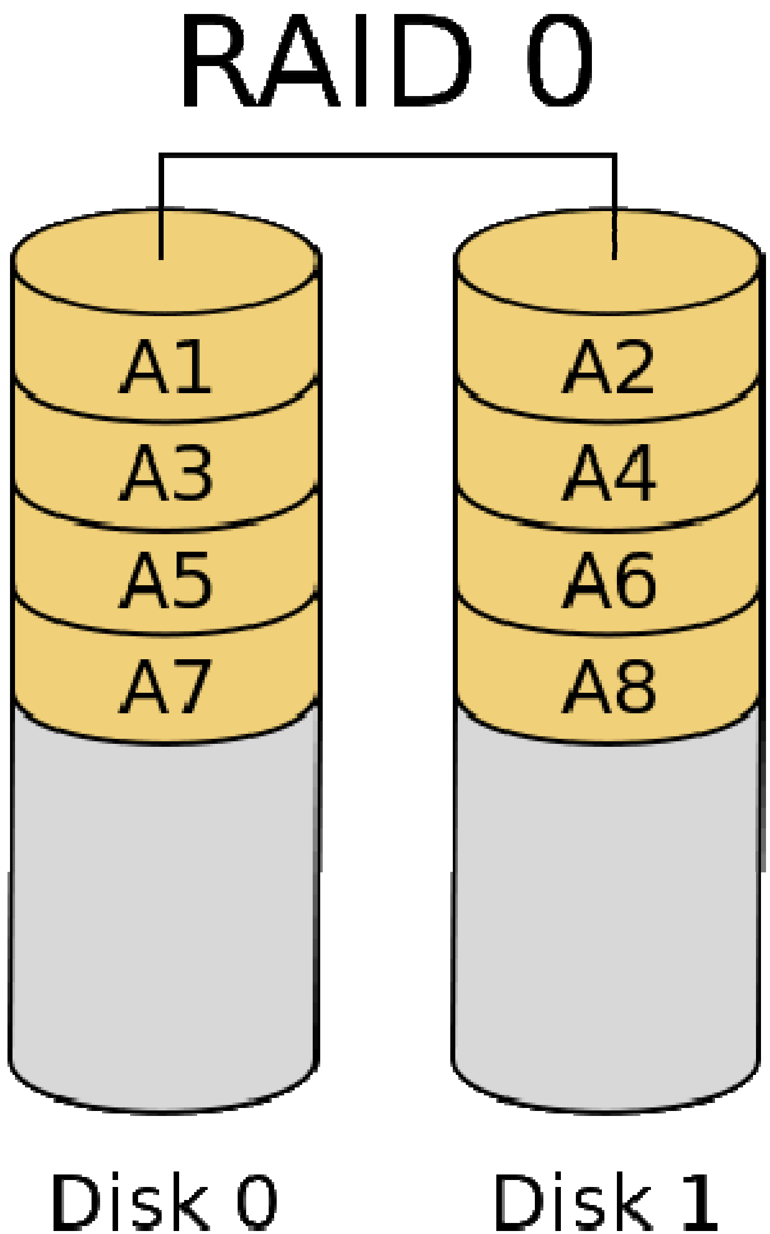
- **R**edundant
- **A**rray of
- ~~**I**nexpensive~~ **I**ndependent ($$$)
- **D**isks

RAID

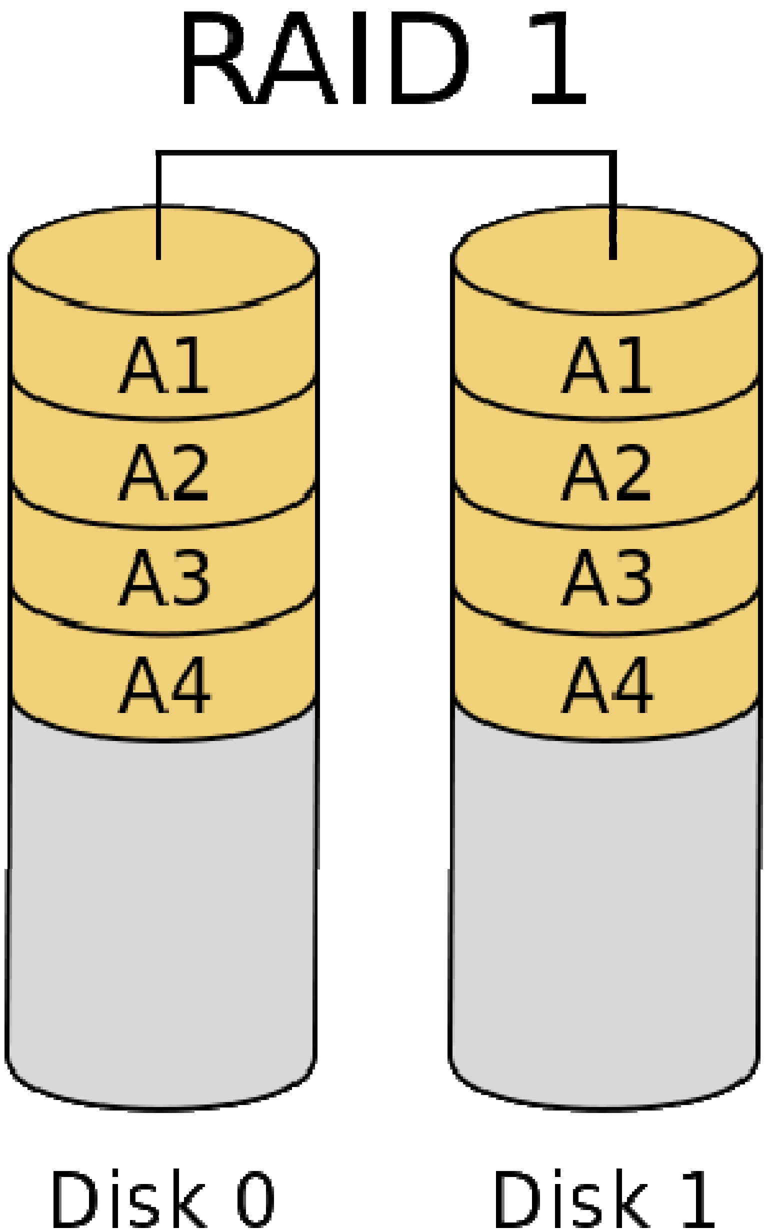
- Create parallelism by splitting (or striping) data across several disk drives

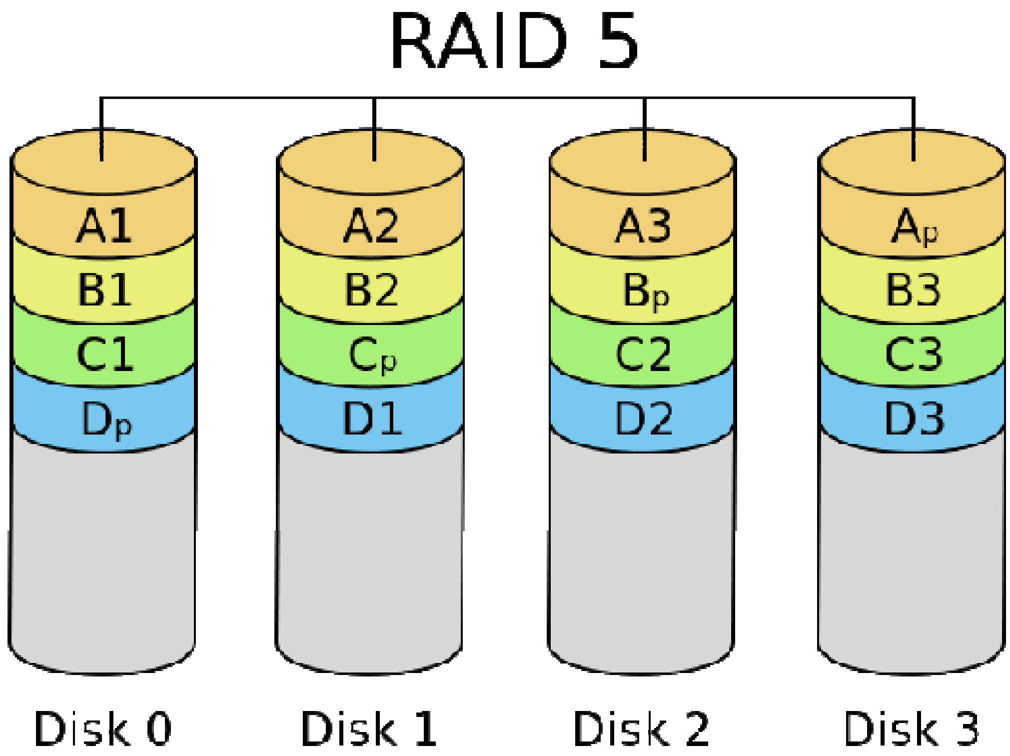
- Add reliability through redundant information of some form or another

- Can be… primitive.



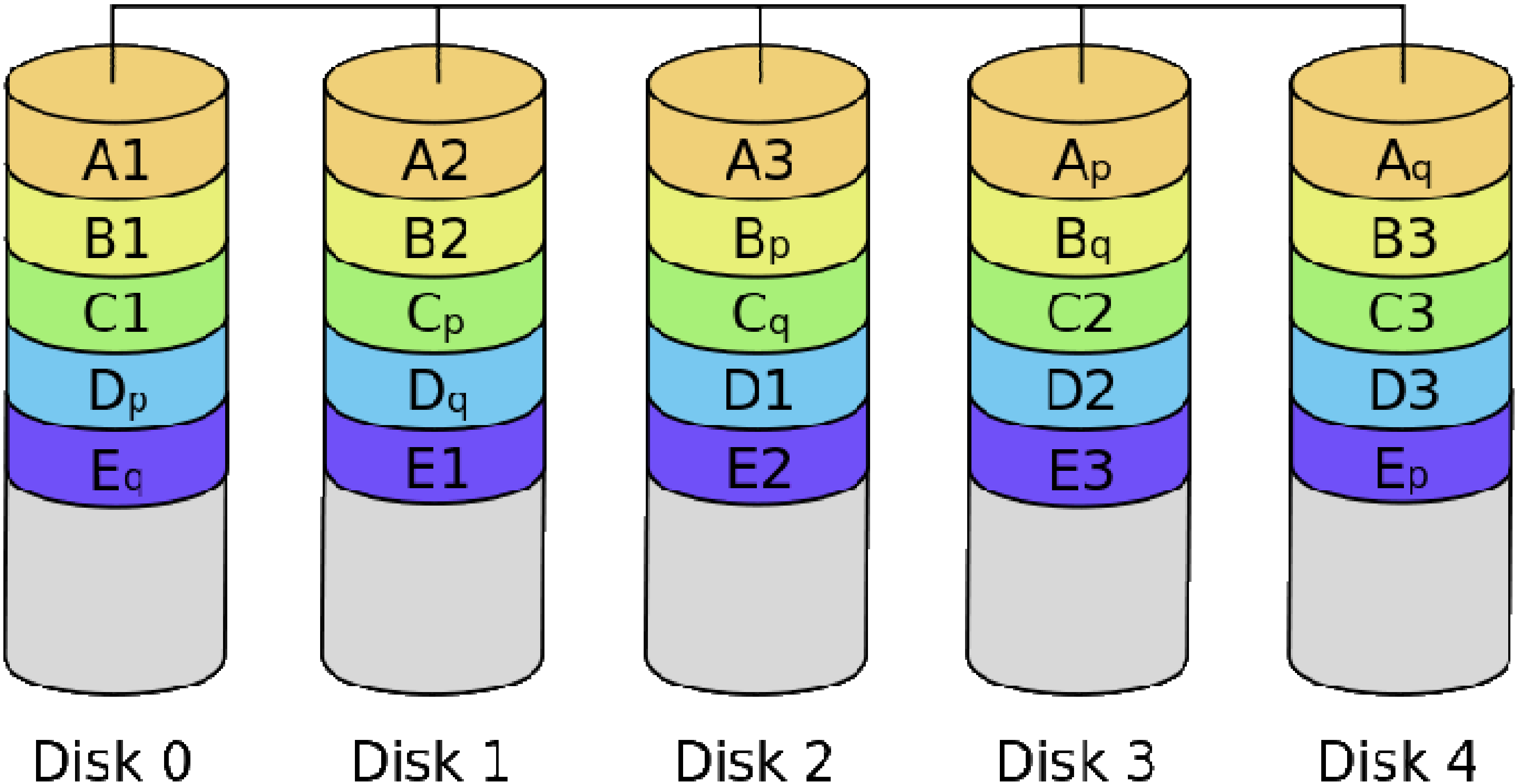
$$\mathrm{MTTF}_{\mathrm{group}} \approx \frac{\mathrm{MTTF}_{\mathrm{disk}}}{\mathrm{number}}$$


RAID 1
A1
A2
A3
A4
A1
A2
A3
A4
Disk 0
Disk 1



- Simple error correcting code scheme based on XOR
  - When Ap XOR A1 XOR A2 XOR A3 = 0, so when any block becomes unknown (via data loss), simply solve for that block

RAID 6



A1
B1
C1
Dp
Eq
A2
B2
Cp
Dq
E1
A3
Bp
Cq
D1
E2
Ap
Bq
C2
D2
E3
Aq
B3
C3
D3
Ep
Disk 0
Disk 1
Disk 2
Disk 3
Disk 4

The General RAID Model

- Each RAID volume has a total of *n* disks
  - *k/n* of the storage is dedicated to clear data
  - *m/n* of the storage is dedicated to coded, redundant data
- Each RAID level is expressed as some *k+m*
  - RAID 0 is *k+0*
  - RAID 1, with *p* replicas, is 1+*p*
  - RAID 5 is *k+1*
  - RAID 6 is *k+2*
- RAID implementations for RAID 6 and above (*m >= 2*) use Reed Solomon coding

Reed-Solomon Codes

Sandia National Laboratories

- Invented by Irving Reed and Gustav Solomon at MIT Lincoln Labs
- Designed for satellite transmissions
  - Symbol-based
  - Errors occur in bursts
  - Error detection and correction
  - Tunable to expected environment
- For our purposes:
  - A symbol is a single byte
  - Each byte is stored on a separate disk
  - We emphasize/optimize for erasure correction

GF(2^8) – Finite Field

- Each byte of stored data represents a member of GF(2^8), as its values range from 0 to 255.
- Addition and subtraction: Bitwise Exclusive OR
  - Thus, the additive identity is the element 00, and these operations are associative and commutative
- Multiplication is accomplished via Boolean polynomial multiplication modulo the specially chosen polynomial $x^8 + x^4 + x^3 + x^2 + 1$
  - Or, for the hardware engineers:
    - $(x \cdot \{02\})_7 = x_6$
    - $(x \cdot \{02\})_6 = x_5$
    - $(x \cdot \{02\})_5 = x_4$
    - $(x \cdot \{02\})_4 = x_3 + x_7$
    - $(x \cdot \{02\})_3 = x_2 + x_7$
    - $(x \cdot \{02\})_2 = x_1 + x_7$
    - $(x \cdot \{02\})_1 = x_0$
    - $(x \cdot \{02\})_0 = x_7$

Information Dispersal

- Generate a matrix of size *(k+m)* x *k* such that any set of *k* rows is linearly independent. One such method:
  - *F[i,j] = i ^ j*
- Use Gaussian elimination on the top *k* rows

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ F_{0,0} & F_{0,1} & F_{0,2} & F_{0,3} \\ F_{1,0} & F_{1,1} & F_{1,2} & F_{1,3} \\ F_{2,0} & F_{2,1} & F_{2,2} & F_{2,3} \\ F_{3,0} & F_{3,1} & F_{3,2} & F_{3,3} \end{pmatrix} \begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{pmatrix} = \begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ p_0 \\ p_1 \\ p_2 \\ p_3 \end{pmatrix}$$

Challenges/Opportunities

- Computational Expense
  - CPUs are not designed to do efficient GF(2^8) multiplication
- Error detection lends itself to inefficient operation
  - Requires another, lower level of error correction
- Failure cases harm performance

- Who says that this has to protect disks?
- Who says there can't be upper levels of RAID?
- Who says we have to replace disks once they fail?
- Who says it's just for data safety?

Thanks.

Acknowledgements: The RAID data organization diagrams were created by Wikipedia user cburnett. These diagrams are licensed under the GNU Free Documentation License, version 1.2 or later.