Betriebssysteme

12. Tutorium - Files and Directories

Peter Bohner

30. Januar 2025

ITEC - Operating Systems Group

Solid-State Drives

NAND based flash memory

Rejoice, TI might be useful once

How long do writes/reads take normally?

NAND based flash memory

Rejoice, TI might be useful once

How long do writes/reads take normally?

- Reading a page: 25µs

NAND based flash memory

Rejoice, TI might be useful once

How long do writes/reads take normally?

- Reading a page: 25µs
- Writing a page: 250µs

NAND based flash memory

Rejoice, TI might be useful once

How long do writes/reads take normally?

- Reading a page: 25µs
- Writing a page: 250µs
- Erasing a block:

NAND based flash memory

Rejoice, TI might be useful once

How long do writes/reads take normally?

- Reading a page: 25µs
- Writing a page: 250µs
- Erasing a block: 2ms

What happens when you just write to a random page?

NAND based flash memory

Rejoice, TI might be useful once

How long do writes/reads take normally?

- Reading a page: 25µs
- Writing a page: 250µs
- Erasing a block: 2ms

What happens when you just write to a random page?

Speeding things up

What could you change so writing pages is faster?

NAND based flash memory

Rejoice, TI might be useful once

How long do writes/reads take normally?

- Reading a page: 25µs
- Writing a page: 250µs
- Erasing a block: 2ms

What happens when you just write to a random page?

Speeding things up

What could you change so writing pages is faster?

- Keep around spare *erased* pages

⇒ You do not pay the erase penalty!

- When do you create / reserve / erase those spare pages?

NAND based flash memory

Rejoice, TI might be useful once

How long do writes/reads take normally?

- Reading a page: 25µs
- Writing a page: 250µs
- Erasing a block: 2ms

What happens when you just write to a random page?

Speeding things up

What could you change so writing pages is faster?

- Keep around spare *erased* pages

⇒ You do not pay the erase penalty!

- When do you create / reserve / erase those spare pages? Probably in the background. Any problems?

NAND based flash memory

Rejoice, TI might be useful once

How long do writes/reads take normally?

- Reading a page: 25µs
- Writing a page: 250µs
- Erasing a block: 2ms

What happens when you just write to a random page?

Speeding things up

What could you change so writing pages is faster?

- Keep around spare *erased* pages

⇒ You do not pay the erase penalty!

- When do you create / reserve / erase those spare pages? Probably in the background. Any problems? Might get exhausted if you write too much data in a short timeframe or the disk is full!

NAND based flash memory

Deleting files

What happens when you delete a file? What effect does that have on the SSD performance?

NAND based flash memory

Deleting files

What happens when you delete a file? What effect does that have on the SSD performance? The block is not freed ⇒ Can't be used as an erased empty page

What can the OS do to combat that?

NAND based flash memory

Deleting files

What happens when you delete a file? What effect does that have on the SSD performance? The block is not freed ⇒ Can’t be used as an erased empty page

What can the OS do to combat that?

The `trim` command

Can be issued by the OS to tell the SSD firmware what pages can be safely erased.

RAID

RAID

What is that?

A **R**edundant **A**rray of **I**ndependent/**I**nexpensive **D**isks

RAID

What is that?

A **R**edundant **A**rray of **I**ndependent/**I**nexpensive **D**isks

Why would you use that?

RAID

What is that?

A **R**edundant **A**rray of **I**ndependent/**I**nexpensive **D**isks

Why would you use that?

- Probably cheaper than a SLED (Single Large Expensive Disk)

RAID

What is that?

A **R**edundant **A**rray of **I**ndependent/**I**nexpensive **D**isks

Why would you use that?

- Probably cheaper than a SLED (Single Large Expensive Disk)
- Might be more resilient

RAID

What is that?

A **R**edundant **A**rray of **I**ndependent/**I**nexpensive **D**isks

Why would you use that?

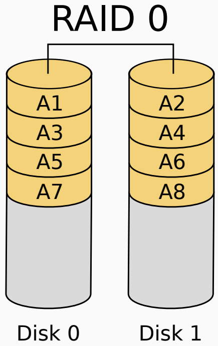
- Probably cheaper than a SLED (Single Large Expensive Disk)
- Might be more resilient
- Might be faster

RAID - And now?

Great, you now have multiple disks. How do you store your files on them?

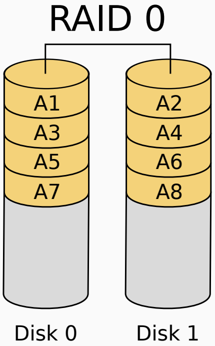
- „I like to live dangerously“ - RAID Level 0
- Mirroring: RAID Level 1
- Historic variants: RAID Level 2 and 3
- Block striping and parity: RAID Level 4
- Block striping and *distributed* parity: RAID Level 5

RAID 0



Benefits / Drawbacks?

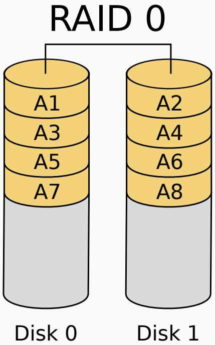
RAID 0



Benefits / Drawbacks?

+ Extremely fast (parallel reads and writes)

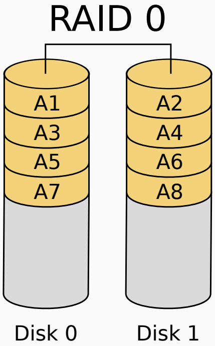
RAID 0



Benefits / Drawbacks?

- + Extremely fast (parallel reads and writes)
- + Can use full capacity

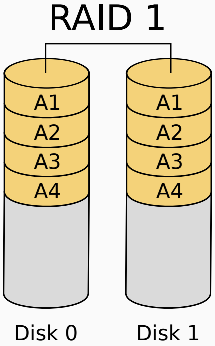
RAID 0



Benefits / Drawbacks?

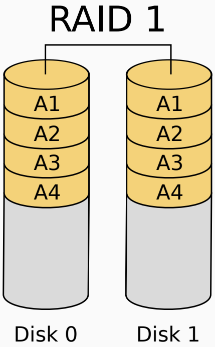
- + Extremely fast (parallel reads and writes)
- + Can use full capacity
- - If a single disk fails your files are toast

RAID 1



Benefits / Drawbacks?

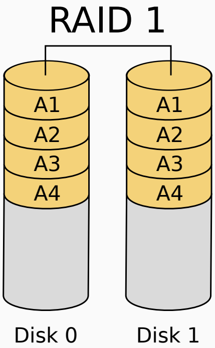
RAID 1



Benefits / Drawbacks?

+ You can lose all but one disk without losing data

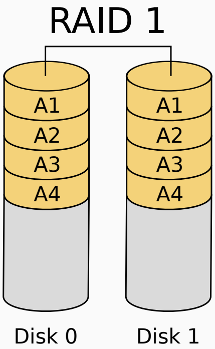
RAID 1



Benefits / Drawbacks?

- + You can lose all but one disk without losing data
- + Parallel reads possible

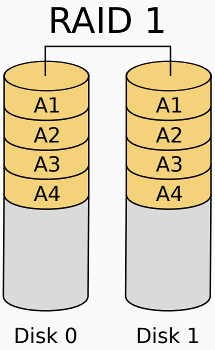
RAID 1



Benefits / Drawbacks?

- + You can lose all but one disk without losing data
- + Parallel reads possible
- – Writes slower as they need to write to all disks

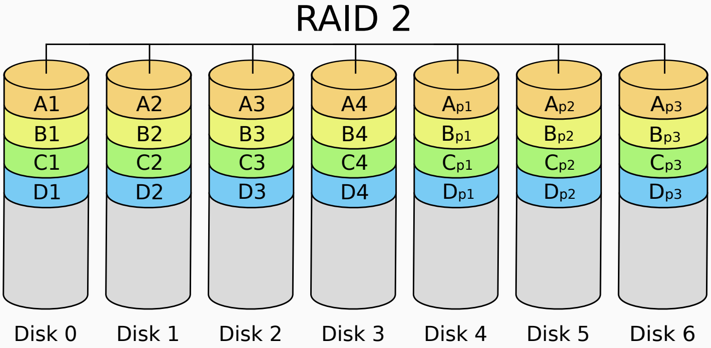
RAID 1



Benefits / Drawbacks?

- + You can lose all but one disk without losing data
- + Parallel reads possible
- – Writes slower as they need to write to all disks
- – Size equals the size of a single disk

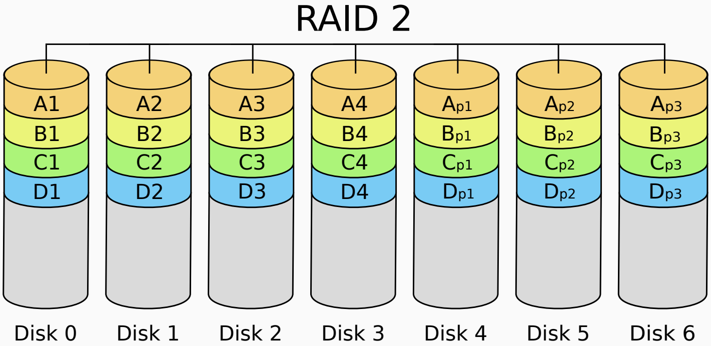
RAID 2



What is that?

- Have $\log_2(N)$ parity disk

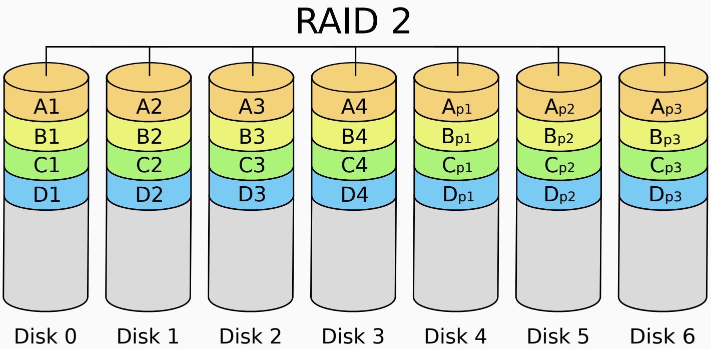
RAID 2



What is that?

- Have $\log_2(N)$ parity disk
- Stripe data at the *bit* level

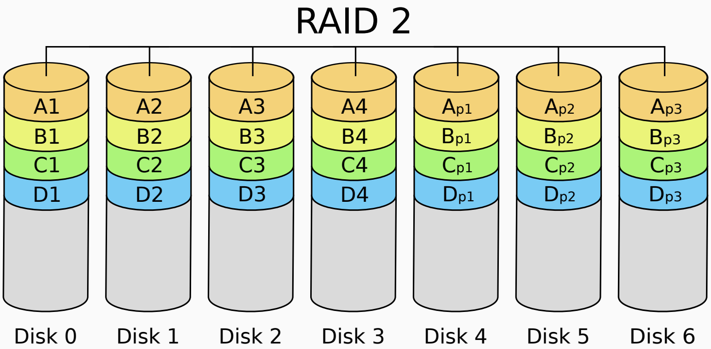
RAID 2



What is that?

- Have $\log_2(N)$ parity disk
- Stripe data at the *bit* level
- Use a hamming code of proper size

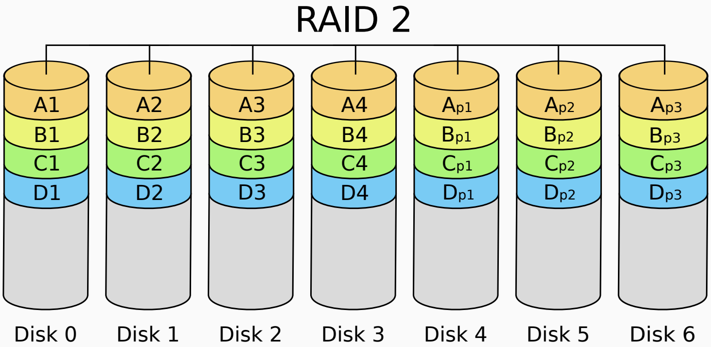
RAID 2



What is that?

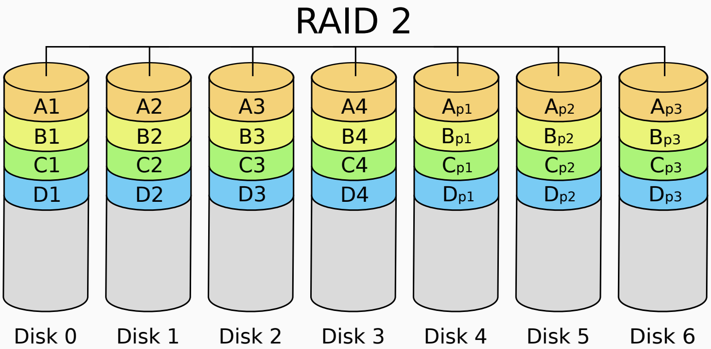
- Have $\log_2(N)$ parity disk
- Stripe data at the *bit* level
- Use a hamming code of proper size
- Spin the disks in lockstep (so you read all bits of your word at once)

RAID 2



Benefits / Drawbacks

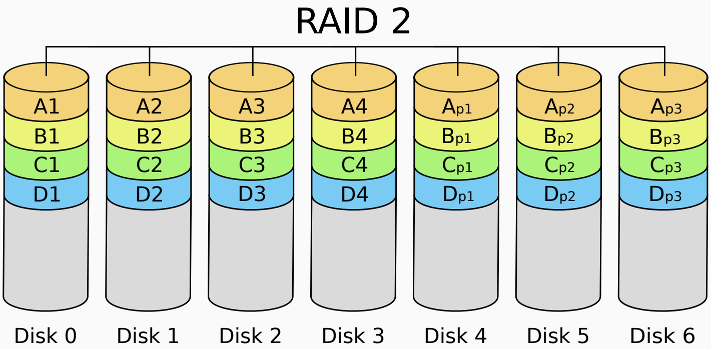
RAID 2



Benefits / Drawbacks

+ External error checking

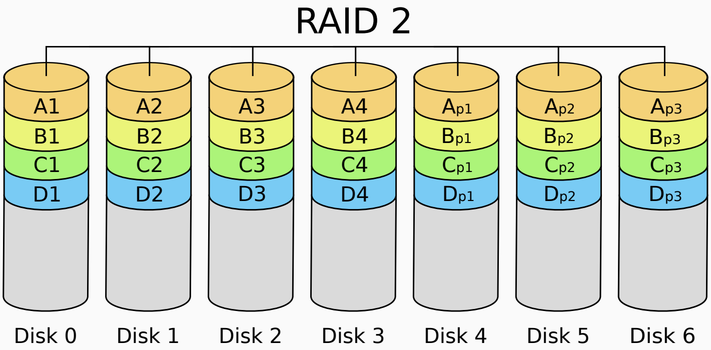
RAID 2



Benefits / Drawbacks

+ External error checking
- Really slow

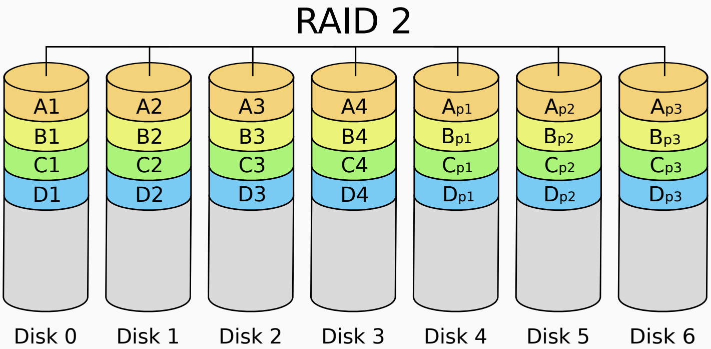
RAID 2



Benefits / Drawbacks

- + External error checking
- – Really slow
- – Not that useful as disks have internal error checking by now

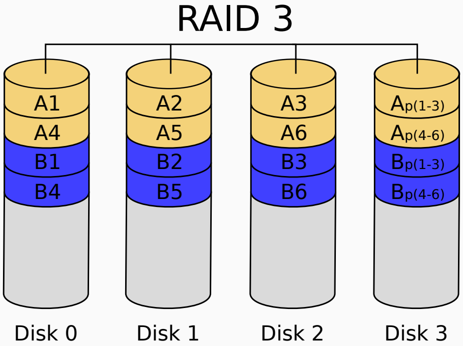
RAID 2



Benefits / Drawbacks

+ External error checking
- Really slow
- Not that useful as disks have internal error checking by now
- Spins in lockstep ⇒ Can only service one request at a time

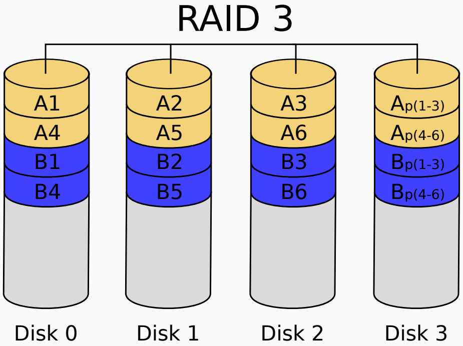
RAID 3



What is that?

- Have a dedicated parity disk

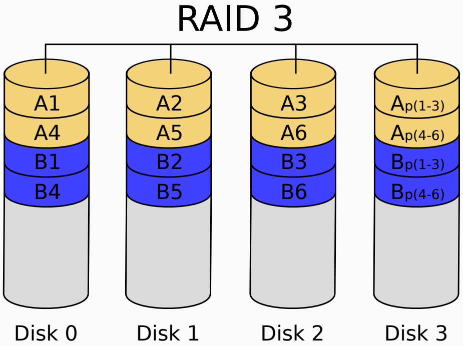
RAID 3



What is that?

- Have a dedicated parity disk
- Stripe data at the *byte* level

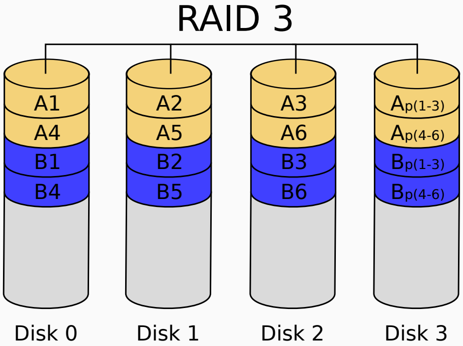
RAID 3



What is that?

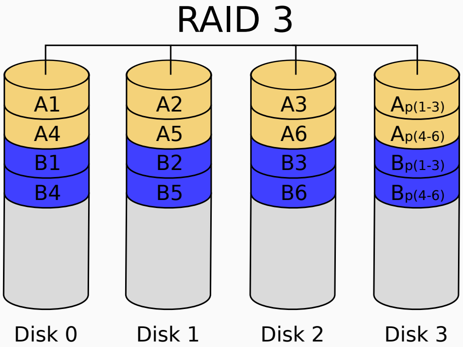
- Have a dedicated parity disk
- Stripe data at the *byte* level
- Spin the disks in lockstep (so you read all bytes of your word at once)

RAID 3



Benefits / Drawbacks

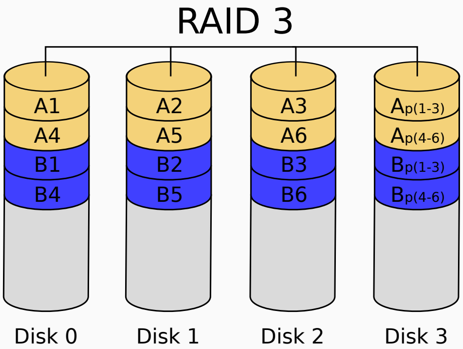
RAID 3



Benefits / Drawbacks

- + You can lose a disk and restore it using the parity

RAID 3

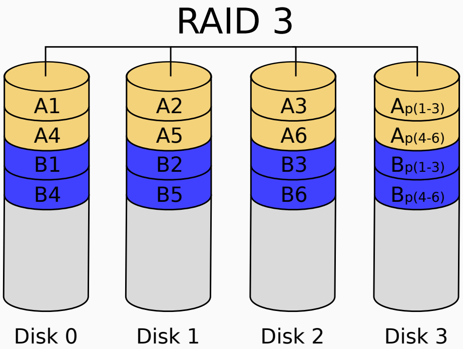


Benefits / Drawbacks

+ You can lose a disk and restore it using the parity

– Slow when reading/writing small files at random locations

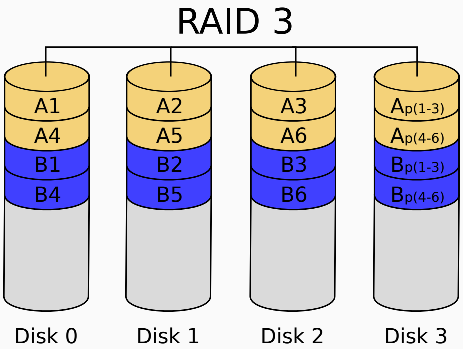
RAID 3



Benefits / Drawbacks

+ You can lose a disk and restore it using the parity
- Slow when reading/writing small files at random locations
- Spins in lockstep ⇒ Can only service one request at a time

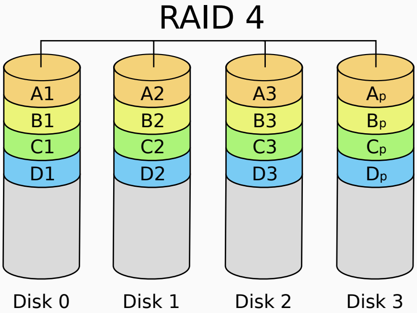
RAID 3



Benefits / Drawbacks

+ You can lose a disk and restore it using the parity
- Slow when reading/writing small files at random locations
- Spins in lockstep ⇒ Can only service one request at a time
- Every write and read hits the same single parity disk

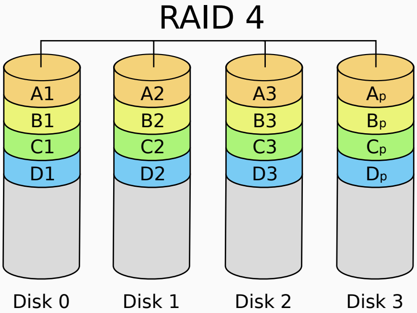
RAID 4



What is that?

- Have a dedicated parity disk

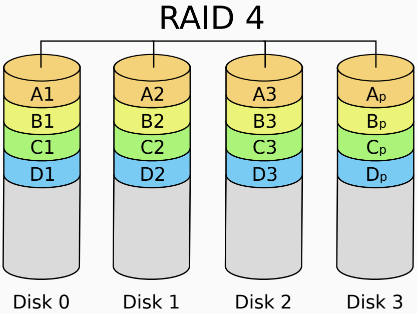
RAID 4



What is that?

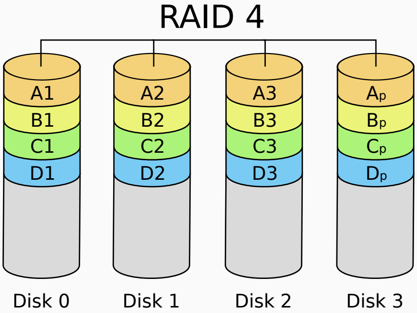
- Have a dedicated parity disk
- Stripe data at the *block* level

RAID 4



Benefits / Drawbacks

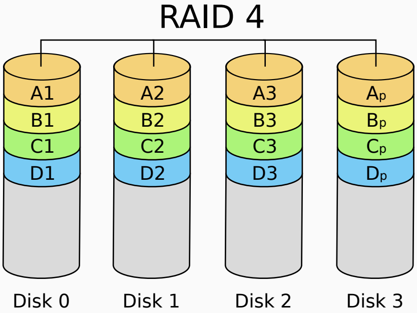
RAID 4



Benefits / Drawbacks

- + You can lose a disk and restore it using the parity

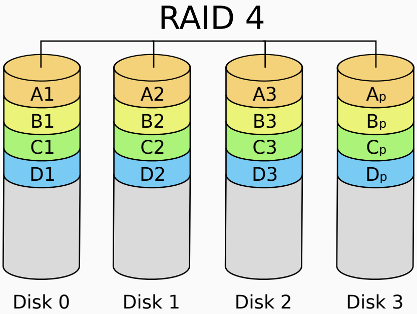
RAID 4



Benefits / Drawbacks

- + You can lose a disk and restore it using the parity
- + Good read performance

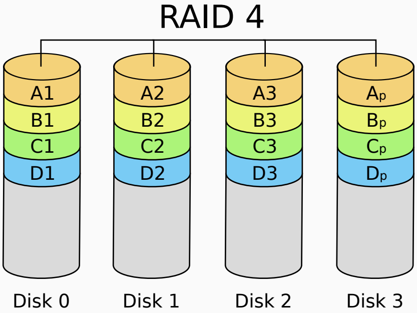
RAID 4



Benefits / Drawbacks

- + You can lose a disk and restore it using the parity
- + Good read performance
- - Every write and read hits the same single parity disk ⇒ Bottleneck, prone to failure

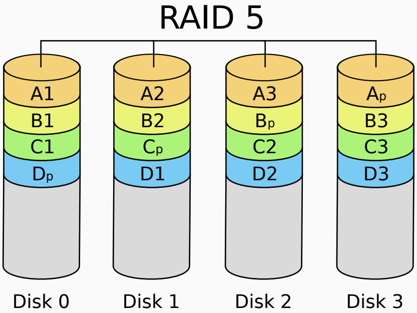
RAID 4



Benefits / Drawbacks

+ You can lose a disk and restore it using the parity
+ Good read performance
- Every write and read hits the same single parity disk ⇒ Bottleneck, prone to failure
- Slow writes (write to same parity disk)

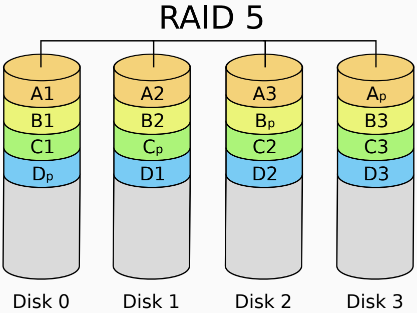
RAID 5



What is that?

- Stripe data at the *block* level

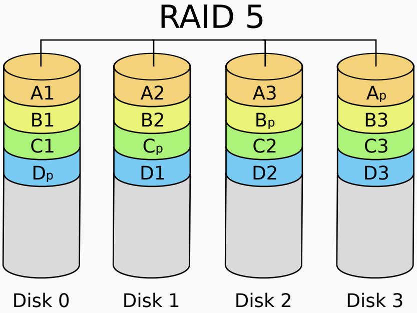
RAID 5



What is that?

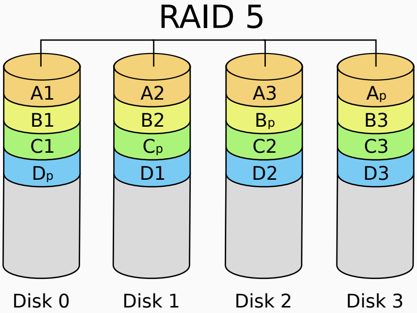
- Stripe data at the *block* level
- Distribute parity across your disks

RAID 5



Benefits / Drawbacks

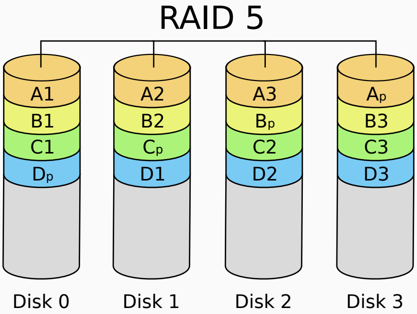
RAID 5



Benefits / Drawbacks

+ You can lose a disk and restore it using the parity

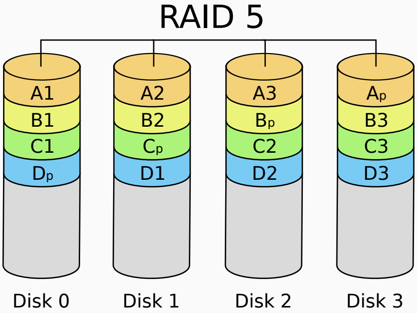
RAID 5



Benefits / Drawbacks

+ You can lose a disk and restore it using the parity
+ Good read performance

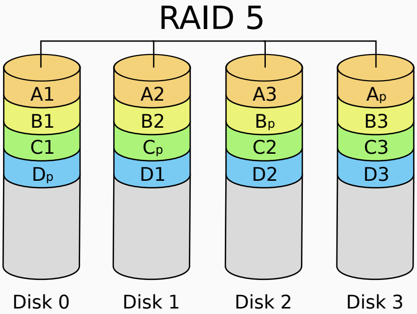
RAID 5



Benefits / Drawbacks

- + You can lose a disk and restore it using the parity
- + Good read performance
- + Okay write performance

RAID 5



Benefits / Drawbacks

+ You can lose a disk and restore it using the parity
+ Good read performance
+ Okay write performance
- Still slower than RAID 0 or a SLED

RAID

Compare SLED and RAID (Level 0, 1, 4, 5)

Each RAID uses 4 disks for actual data storage.

How many disks do you need?

- SLED: 1

RAID

Compare SLED and RAID (Level 0, 1, 4, 5)

Each RAID uses 4 disks for actual data storage.

How many disks do you need?

- SLED: **1**
- RAID 0: **4**

RAID

Compare SLED and RAID (Level 0, 1, 4, 5)

Each RAID uses 4 disks for actual data storage.

How many disks do you need?

- SLED: 1
- RAID 0: 4
- RAID 1: 8

RAID

Compare SLED and RAID (Level 0, 1, 4, 5)

Each RAID uses 4 disks for actual data storage.

How many disks do you need?

- SLED: **1**
- RAID 0: **4**
- RAID 1: **8**
- RAID 4: **5**

RAID

Compare SLED and RAID (Level 0, 1, 4, 5)

Each RAID uses 4 disks for actual data storage.

How many disks do you need?

- SLED: **1**
- RAID 0: **4**
- RAID 1: **8**
- RAID 4: **5**
- RAID 5: **5**

RAID

You want to modify one byte of data. How many blocks do you need to read/write?

- SLED: 1 read + 1 write

RAID

You want to modify one byte of data. How many blocks do you need to read/write?

- SLED: 1 read + 1 write
- RAID 0: 1 read + 1 write

RAID

You want to modify one byte of data. How many blocks do you need to read/write?

- SLED: 1 read + 1 write
- RAID 0: 1 read + 1 write
- RAID 1: 1 read + 2 write (1 data + 1 mirror)

RAID

You want to modify one byte of data. How many blocks do you need to read/write?

- SLED: 1 read + 1 write
- RAID 0: 1 read + 1 write
- RAID 1: 1 read + 2 write (1 data + 1 mirror)
- RAID 4: 2 read (data + old parity) + 2 write (data + new parity)

RAID

You want to modify one byte of data. How many blocks do you need to read/write?

- SLED: 1 read + 1 write
- RAID 0: 1 read + 1 write
- RAID 1: 1 read + 2 write (1 data + 1 mirror)
- RAID 4: 2 read (data + old parity) + 2 write (data + new parity)
- RAID 5: 2 read (data + old parity) + 2 write (data + new parity)

RAID vs. Backup

You are using RAID

- You accidentally delete a file.

RAID vs. Backup

You are using RAID

- You accidentally delete a file. ***GONE***
- You accidentally overwrite a file.

RAID vs. Backup

You are using RAID

- You accidentally delete a file. ***GONE***
- You accidentally overwrite a file. ***GONE***
- Some data gets corrupted on one disk.

RAID vs. Backup

You are using RAID

- You accidentally delete a file. ***GONE***
- You accidentally overwrite a file. ***GONE***
- Some data gets corrupted on one disk. ***GONE*** *(probably)*
- The poor intern connects to the production database. (Or here)

RAID vs. Backup

You are using RAID

- You accidentally delete a file. ***GONE***
- You accidentally overwrite a file. ***GONE***
- Some data gets corrupted on one disk. ***GONE*** *(probably)*
- The poor intern connects to the production database. (Or here) ***GONE***
- A crypto-locker takes out your computer.

RAID vs. Backup

You are using RAID

- You accidentally delete a file. ***GONE***
- You accidentally overwrite a file. ***GONE***
- Some data gets corrupted on one disk. ***GONE*** *(probably)*
- The poor intern connects to the production database. (Or here) ***GONE***
- A crypto-locker takes out your computer. Believe it or not, ~~***JAIL***~~ ***GONE***

RAID vs. Backup

You are using RAID

- You accidentally delete a file. ***GONE***
- You accidentally overwrite a file. ***GONE***
- Some data gets corrupted on one disk. ***GONE*** *(probably)*
- The poor intern connects to the production database. (Or here) ***GONE***
- A crypto-locker takes out your computer. Believe it or not, *~~JAIL~~ GONE*

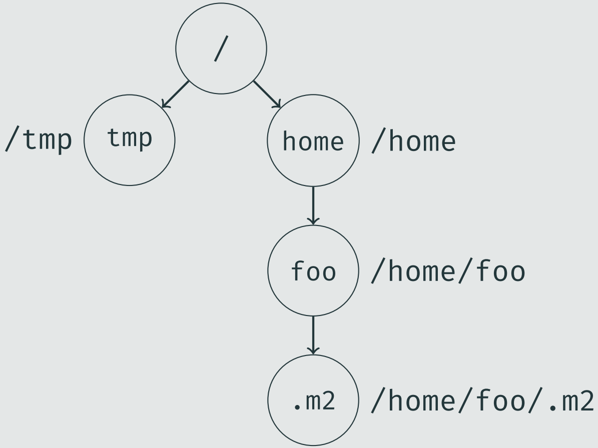
So what do we learn?

RAID IS NO SUBSTITUTE FOR A BACKUP

Files And Directories

Files And Directories

Paths (Linux)



Reading A File

What are the two basic access methods (patterns) for reading a file?

- Sequential Access

  Accessed in order, reading sequential bytes. Writes append at the end.

- Random Access

  Reading and writing at arbitrary positions, programmer needs to specify where to write/read

Which Library Functions and System Calls do we have to access files?

Opening the file

Which Library Functions and System Calls do we have to access files?

Opening the file

`fopen` / `open`. Provide mode as argument (read/write, where, create, etc.)

Which Library Functions and System Calls do we have to access files?

Opening the file

`fopen` / `open`. Provide mode as argument (read/write, where, create, etc.)

Closing a file

Which Library Functions and System Calls do we have to access files?

Opening the file

`fopen` / `open`. Provide mode as argument (read/write, where, create, etc.)

Closing a file

`fclose` / `close`

Which Library Functions and System Calls do we have to access files?

Opening the file

`fopen` / `open`. Provide mode as argument (read/write, where, create, etc.)

Closing a file

`fclose` / `close`

Reading from a file

Which Library Functions and System Calls do we have to access files?

Opening the file

`fopen` / `open`. Provide mode as argument (read/write, where, create, etc.)

Closing a file

`fclose` / `close`

Reading from a file

`fread` / `read`

Which Library Functions and System Calls do we have to access files?

Opening the file

`fopen` / `open`. Provide mode as argument (read/write, where, create, etc.)

Closing a file

`fclose` / `close`

Reading from a file

`fread` / `read`

Writing to a file

Which Library Functions and System Calls do we have to access files?

Opening the file

`fopen` / `open`. Provide mode as argument (read/write, where, create, etc.)

Closing a file

`fclose` / `close`

Reading from a file

`fread` / `read`

Writing to a file

`fwrite` / `write`

Which Library Functions and System Calls do we have to access files?

Opening the file

`fopen` / `open`. Provide mode as argument (read/write, where, create, etc.)

Closing a file

`fclose` / `close`

Reading from a file

`fread` / `read`

Writing to a file

`fwrite` / `write`

Flushing dirty buffers

Which Library Functions and System Calls do we have to access files?

Opening the file

`fopen` / `open`. Provide mode as argument (read/write, where, create, etc.)

Closing a file

`fclose` / `close`

Reading from a file

`fread` / `read`

Writing to a file

`fwrite` / `write`

Flushing dirty buffers

The library functions sometimes buffer to reduce the amount of syscalls.

`fflush` flushes those buffers.

Random Access

Which system calls / library functions move the „current position pointer“?

Random Access

Which system calls / library functions move the „current position pointer“?

- Linux: `fseek` / `lseek`
- Windows: `SetFilePointerEx` / `SetFilePointer`

Random Access

Which system calls / library functions move the „current position pointer“?

- Linux: `fseek` / `lseek`
- Windows: `SetFilePointerEx` / `SetFilePointer`

What happens when you move the cursor behind the end of the file?

Random Access

Which system calls / library functions move the „current position pointer“?

- Linux: `fseek` / `lseek`
- Windows: `SetFilePointerEx` / `SetFilePointer`

What happens when you move the cursor behind the end of the file?

It creates a hole filled with zeros!

Random Access

Which system calls / library functions move the „current position pointer“?

- Linux: `fseek` / `lseek`
- Windows: `SetFilePointerEx` / `SetFilePointer`

What happens when you move the cursor behind the end of the file?

It creates a hole filled with zeros! Such a file is called a *sparse* file and some file systems might not store empty regions.

Random Access

How could you implement random access without a dedicated file pointer?

Add an offset to the `read` and `write` system call.

Random Access

How could you implement random access without a dedicated file pointer?

Add an offset to the `read` and `write` system call.

- + Save a system call

Random Access

How could you implement random access without a dedicated file pointer?

Add an offset to the `read` and `write` system call.

+ Save a system call
- Caller need to keep track of the current offset for sequential reads

Random Access

How could you implement random access without a dedicated file pointer?

Add an offset to the `read` and `write` system call.

- + Save a system call
- - Caller need to keep track of the current offset for sequential reads

`mmap` the file (works quite well, might cause page faults. Probably faster for large files than read/write calls)

Walking a directory

What system calls do you need to list files in a Linux directory?

Walking a directory

What system calls do you need to list files in a Linux directory?

1. `opendir`

Walking a directory

What system calls do you need to list files in a Linux directory?

1. `opendir`
2. `readdir`:

Walking a directory

What system calls do you need to list files in a Linux directory?

1. `opendir`
2. `readdir`: Returns a `dirent` with a *relative path*

Walking a directory

What system calls do you need to list files in a Linux directory?

1. `opendir`
2. `readdir`: Returns a `dirent` with a *relative path*
3. `closedir`

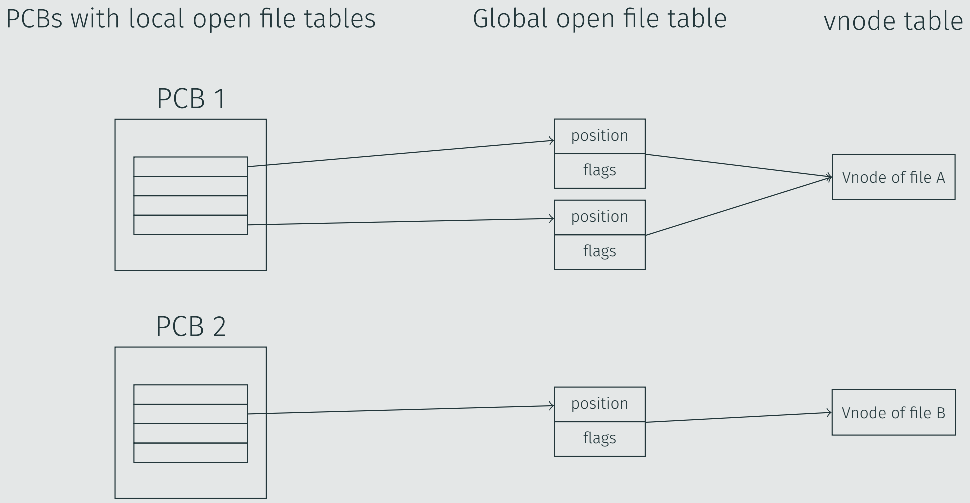
Open Files

Kernel Data Structures For Open Files

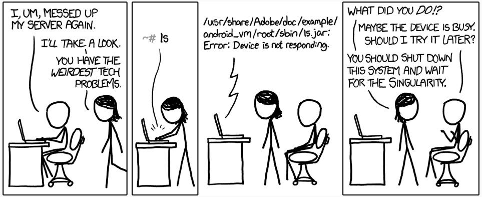
What structures does the kernel use for open files?

Kernel Data Structures For Open Files

What structures does the kernel use for open files?



Ende


I, UM, MESSED UP MY SERVER AGAIN.
I'LL TAKE A LOOK.
YOU HAVE THE WEIRDEST TECH PROBLEMS.
~# ls
/usr/share/Adobe/doc/example/android_vm/root/sbin/ls.jar: Error: Device is not responding.
WHAT DID YOU DO!?
MAYBE THE DEVICE IS BUSY. SHOULD I TRY IT LATER?
YOU SHOULD SHUT DOWN THIS SYSTEM AND WAIT FOR THE SINGULARITY.

XKCD 1084 - Server Problem

FRAGEN?

https://forms.gle/9CwJSKidKibubran9

Bis nächste Woche