Operating Systems

Tutorial 1

Michael Tänzer

os-tut@nhng.de
http://os-tut.nhng.de

1st February 2011

Outline

True or False

- When using linked allocation files can only be accessed sequentially
- When using inodes it doesn't matter whether blocks are allocated contiguously or not
- The file size is stored in the inode

True or False

- When using linked allocation files can only be accessed sequentially
- When using inodes it doesn't matter whether blocks are allocated contiguously or not
- The file size is stored in the inode

True or False

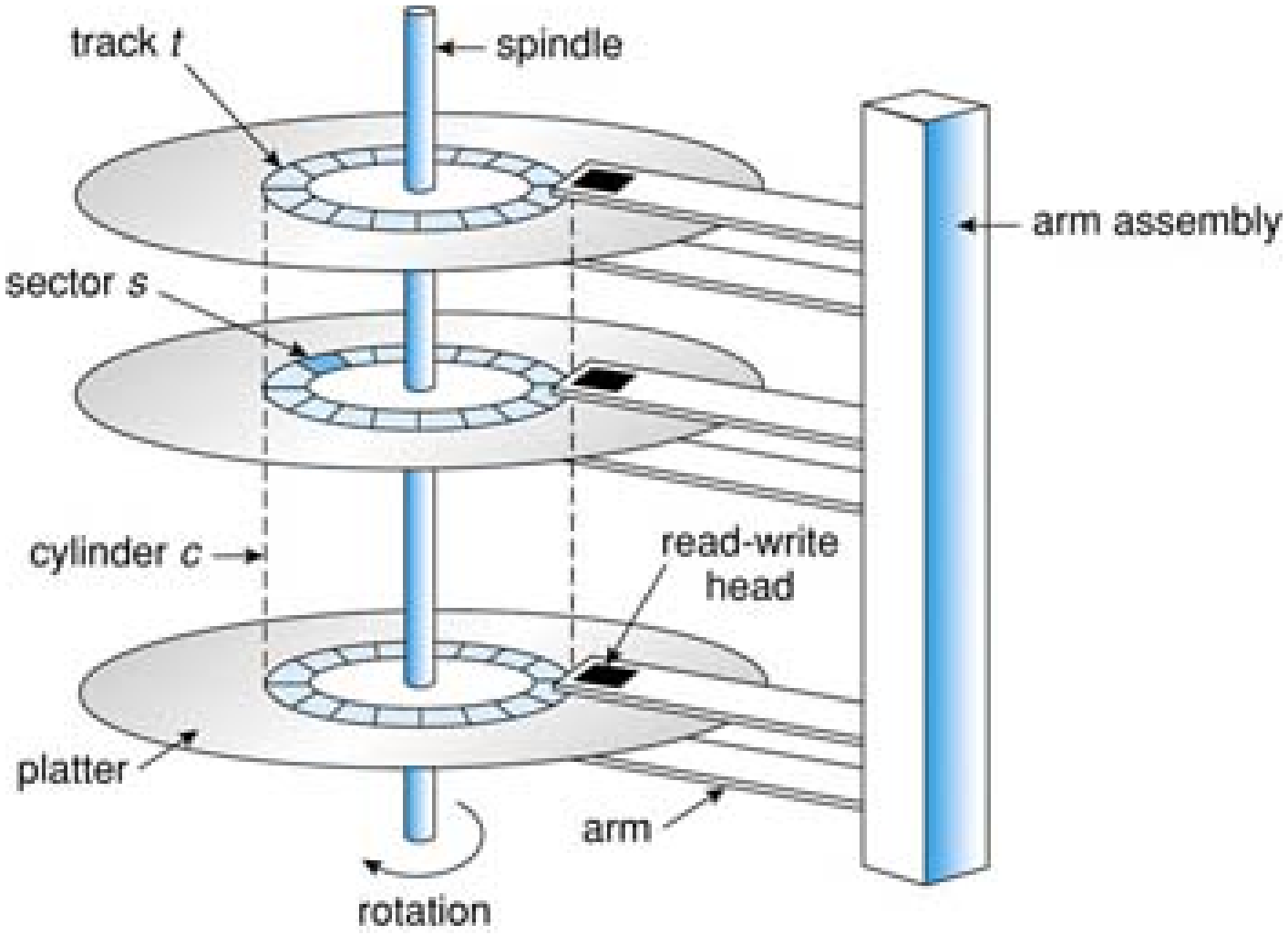
- When using linked allocation files can only be accessed sequentially
- When using inodes it doesn't matter whether blocks are allocated contiguously or not
- The file size is stored in the inode

True or False

- When using linked allocation files can only be accessed sequentially
- When using inodes it doesn't matter whether blocks are allocated contiguously or not
- The file size is stored in the inode

Explain the terms cylinder, track and sector

Explain the terms cylinder, track and sector



Estimate the sustained transfer rate

Ignore the time to move to the next track and assume no initial seek is required

Hard Disk

- 7200 RPM
- 512 bytes sector size
- 160 sectors per track

Estimate the sustained transfer rate

Ignore the time to move to the next track and assume no initial seek is required

Hard Disk

- 7200 RPM
- 512 bytes sector size
- 160 sectors per track

$$7200 \text{ RPM} = \frac{7200}{60} \text{ rounds/s} = 120 \text{ tracks/s}$$

$$1 \text{ track} = 160 \text{ sectors} \cdot 512 \text{ bytes/sector} = 81920 \text{ bytes}$$

$$\Rightarrow \text{transfer rate} = 120 \text{ tracks/s} \cdot 81920 \text{ bytes/track} = 9600 \text{ KB/s}$$

What is track skewing?

What is track skewing?

- If tracks are aligned seeking to the next track would miss the first sector (as the disk is still spinning while the arm moves)

$\Rightarrow$ Each track has a little offset to the previous one

Is the mean transfer rate affected by track skewing when seeking to another track than the adjacent one?

Assume large amounts of data are read

Is the mean transfer rate affected by track skewing when seeking to another track than the adjacent one?

Assume large amounts of data are read

- Rotational delay generally doesn't fit the track skew when not seeking to the next track
- But no need to wait for the first sector, other sectors can be cached as they are probably needed later

$\Rightarrow$ No effective impact on throughput, but on latency

How is data on hard drives addressed?

How is data on hard drives addressed?

CHS Specify **C**ylinder, **H**ead and **S**ector of a block of data (usually 512 bytes). Limited to just under 8 GB (1024 cylinders, 255 heads, 63 sectors)

LBA **L**ogical **B**lock **A**ddressing uses simple linear addresses for the blocks

Explain the term sector sparing

What problem can occur when it's used and how can it be mitigated?

Explain the term sector sparing

What problem can occur when it's used and how can it be mitigated?

- Disk contains spare sectors which are hidden to the OS
- When a defective sector is detected the disk controller replaces the bad sector with a spare

$\Rightarrow$ Future requests to the bad sector are redirected to the spare

- The mapping is transparent to the OS

Explain the term sector sparing

What problem can occur when it's used and how can it be mitigated?

- Disk contains spare sectors which are hidden to the OS
- When a defective sector is detected the disk controller replaces the bad sector with a spare

⇒ Future requests to the bad sector are redirected to the spare

- The mapping is transparent to the OS

– Real structure differs from the structure the OS 'sees'

⇒ The disk scheduler of the OS could make a decision which would be good in theory but is far from optimal in reality

- One could have some spare sectors on each track so the difference doesn't become very big

Explain the term sector slipping

Explain the term sector slipping

- Similar to sector sparing
- Instead of only remapping the bad sector to the spare one all sectors behind the bad sector are remapped one spot (until the cascade reachs a spare sector)

$\Rightarrow$ The bad sector is mapped to the sector directly behind it

$+$ The difference between the abstract disk layout and the real one is only one sector offset

What is Native Command Queuing (NCQ)?

What is Native Command Queuing (NCQ)?

- When multiple requests have to be handled they might be reordered to improve seek times
- Traditionally a task of the OS
- Real hard drive layout differed more and more to the layout assumed by the OS

$\Rightarrow$ OS can't make a good decision on what might be a good command schedule

$\Rightarrow$ Put the reordering strategy into the hard disk firmware

When writing byte-granular data to a file is it reasonable to assume the data is physically on the disk after `write`?

When writing byte-granular data to a file is it reasonable to assume the data is physically on the disk after `write`?

- If the block was flushed to disk after each byte, performance would be very poor

$\Rightarrow$ Writes are cached, i. e. not immediately physically written to the disk

What is the basic idea of a file system cache?

What is the basic idea of a file system cache?

- Recently referenced blocks are likely to be accessed again in the near future

⇒ Keep a copy instead of querying the disk multiple times for the same block

- Drop the copy if it is not likely to be used anymore or if the cache is full
- Explicit calls (`flush`, `sync`) or a daemon (`flushd`) may write modified copies back to disk (that doesn't necessarily mean the copy is dropped)

Fixed vs. variable limit between file system cache and main memory used for paging

Fixed vs. variable limit between file system cache and main memory used for paging

Fixed Limit

+ Easy to implement
+ Possible to give guarantees

Variable Limit

+ Adaptable to different workloads (much file I/O vs. high memory use)

Pros and Cons: Read Ahead

Pros and Cons: Read Ahead

+ Good performance on sequential/predictable file access patterns
+ Improved disk throughput (if files in successive blocks)
– Wasted bandwidth if data is not used

What is swapping?

- Traditionally: pause whole processes and move their memory resident image to disk
- With paging: Store single pages on disk, evicting them from memory

Swap space in file vs. separate partition

Swap space in file vs. separate partition

Swap File

+ Can be accessed like a normal file ⇒ easier to implement
+ Can grow and shrink on demand
– Each access is subject to the normal file operations ⇒ more overhead
– Might get fragmented (especially if size is dynamic)

Swap Partition

+ Raw block access possible ⇒ less overhead
+ Data placement in the partition can be optimized for speed (no safety needed)
– Fixed size

What's anonymous memory?

Why can non-anonymous memory be handled differently with respect to swapping?

What's anonymous memory?

Why can non-anonymous memory be handled differently with respect to swapping?

- Anonymous memory are those memory regions which weren't directly loaded from a file on the file system (i. e. stack, heap and uninitialised data)
- Non-anonymous memory is associated with a file (e. g. the application's binary, a library, a memory mapped file)

$\Rightarrow$ If a non-anonymous page is chosen for eviction it doesn't need to be swapped out to the global swap area but the associated file can serve as swap area

- Exception: modified code (binaries and libraries) should not (and probably can't due to missing privileges) be written back to the original file but to the global swap area

Questions & Comments

Any questions or comments?

Prepare Questions for the Next Tutorial

- Next week will be plenty of room for asking questions

⇒ If there's anything that needs some more explanation, please tell me

- If you send me questions beforehand (`os-tut@nhng.de`) I might even be able to do some research

The End


CLEVERLY, THE E.U. OFFSETS ITS BUDGET DEFECIT FOR 2009
intel's MONEY inside
COPYRIGHT©2009 J.D. "Illiad" Frazer HTTP://WWW.USERFRIENDLY.ORG/
UserFriendly.Org