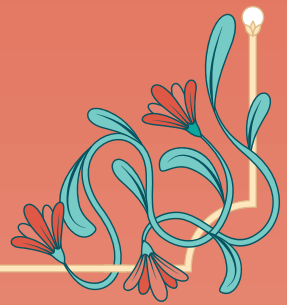
THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
EUROPE

Embedded Linux Conference

Caching in the Storage Path

How to gain performance without loosing data



THE LINUX FOUNDATION

Hi, I’m Werner, I’m …

… using Linux since Fedora 1

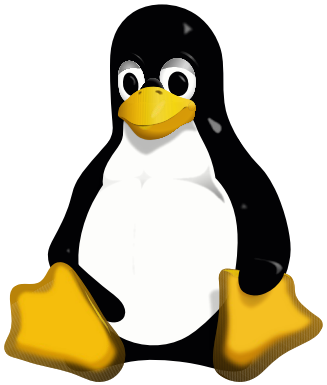
… spent 2004-2005 @IBM

… @Thomas-Krenn since 2005

… trying to bring Linux to
young people
(joined board @LPI for that)



Some questions...

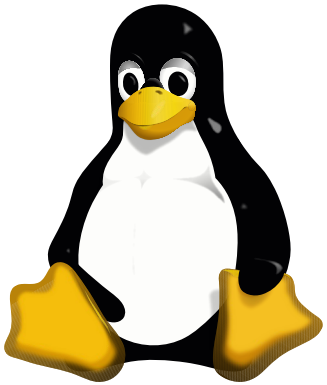




Some questions...

Have you already ...

... run **tune2fs -l**?

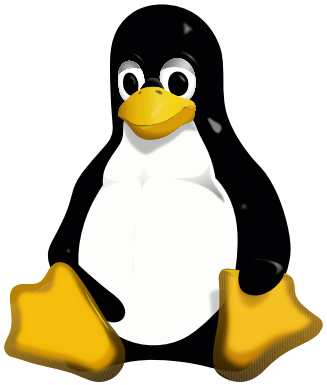


Some questions...

Have you already ...

... run **tune2fs -l**?

... run **hdparm**, **sdparm**, or **nvme-cli**?





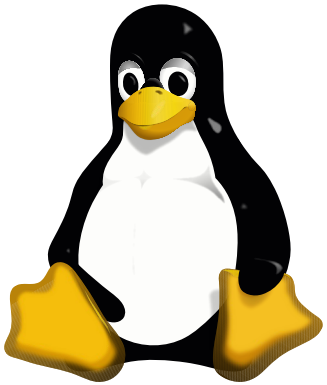
Some questions...

Have you already ...

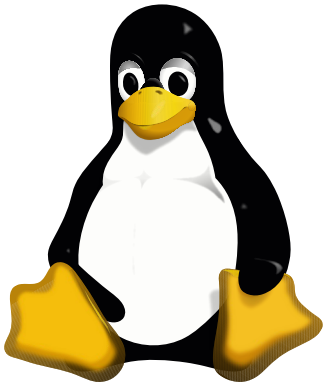
... run **tune2fs -l**?

... run **hdparm**, **sdparm**, or **nvme-cli**?

... run **stat**?



Some questions...



Have you already ...

... run **tune2fs -l**?

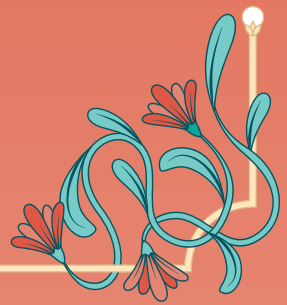
... run **hdparm**, **sdparm**, or **nvme-cli**?

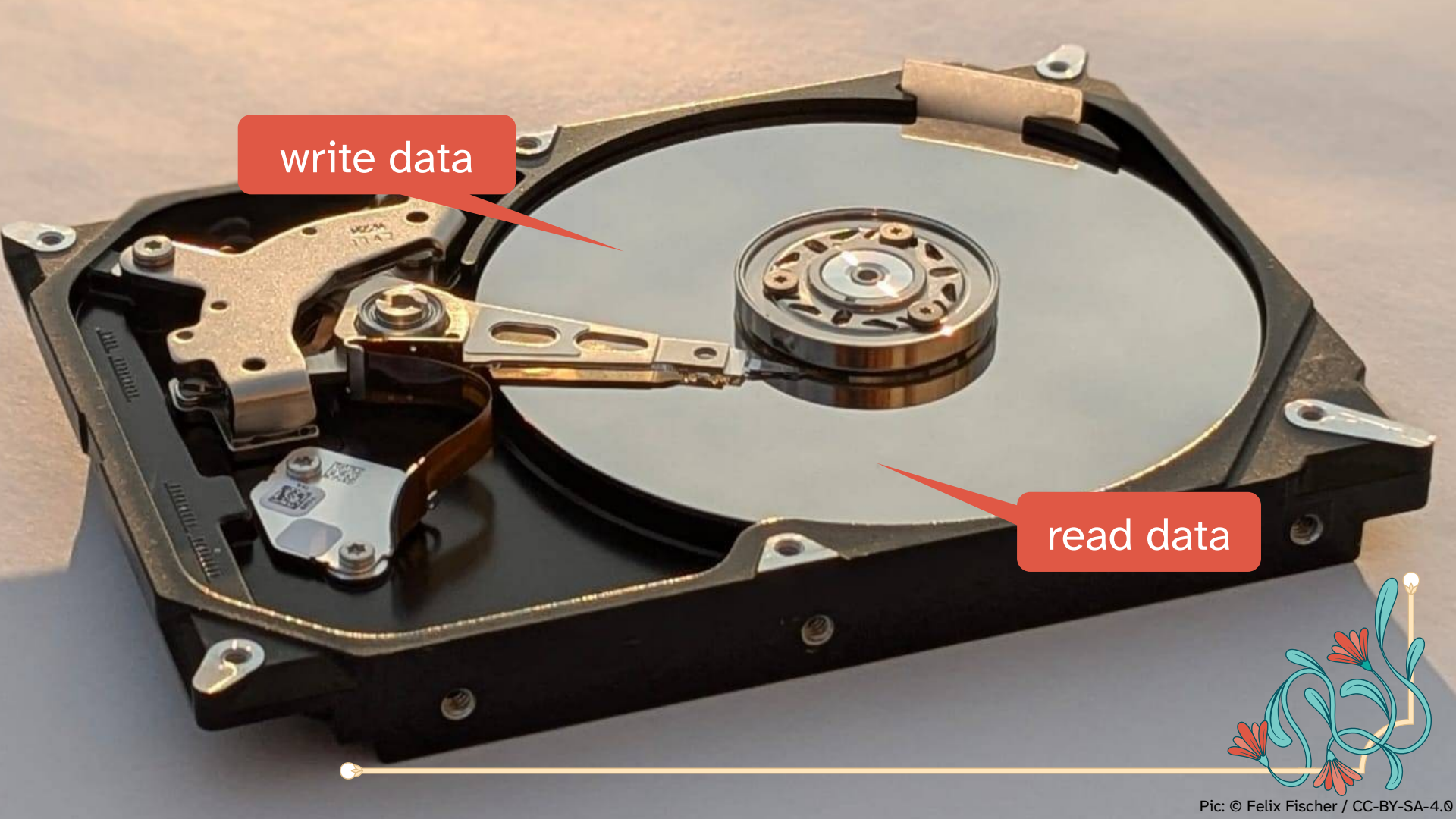
... run **stat**?

... explained fs **“write barriers”** to someone?

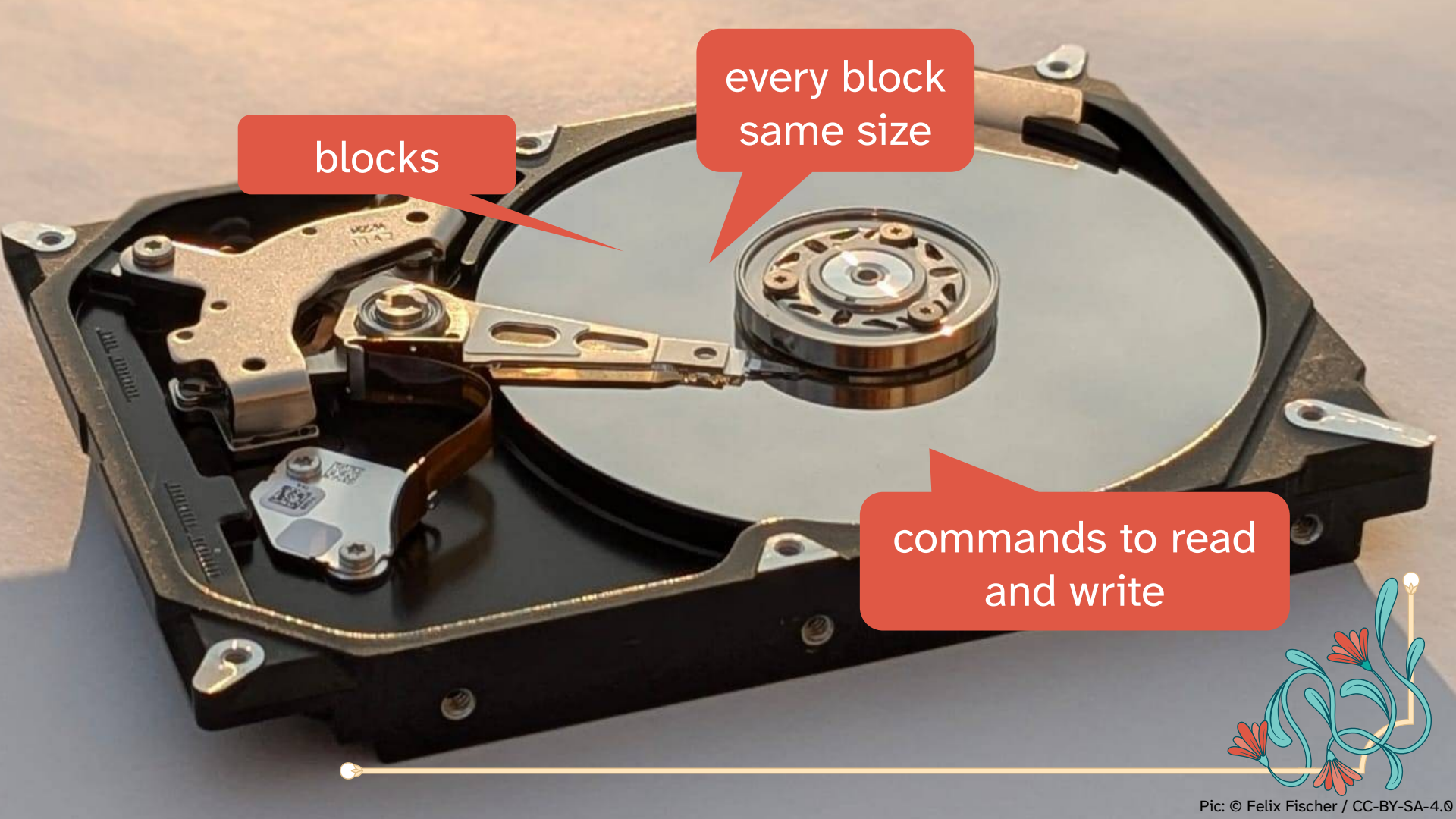
THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
EUROPE
Embedded Linux Conference

Storage – the user's view





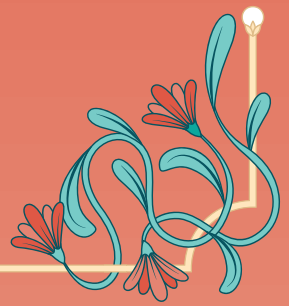
Pic: © Felix Fischer / CC-BY-SA-4.0

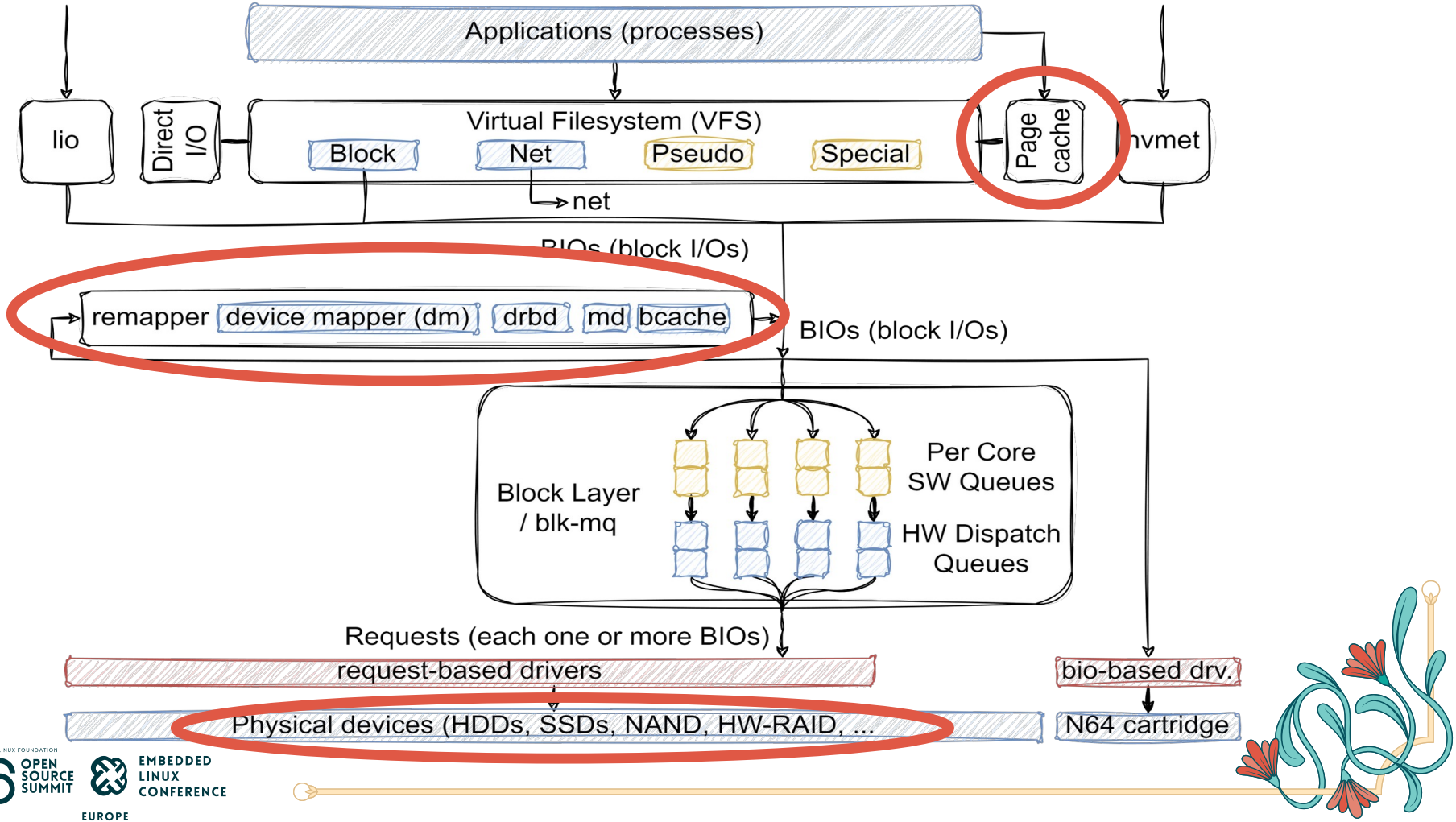


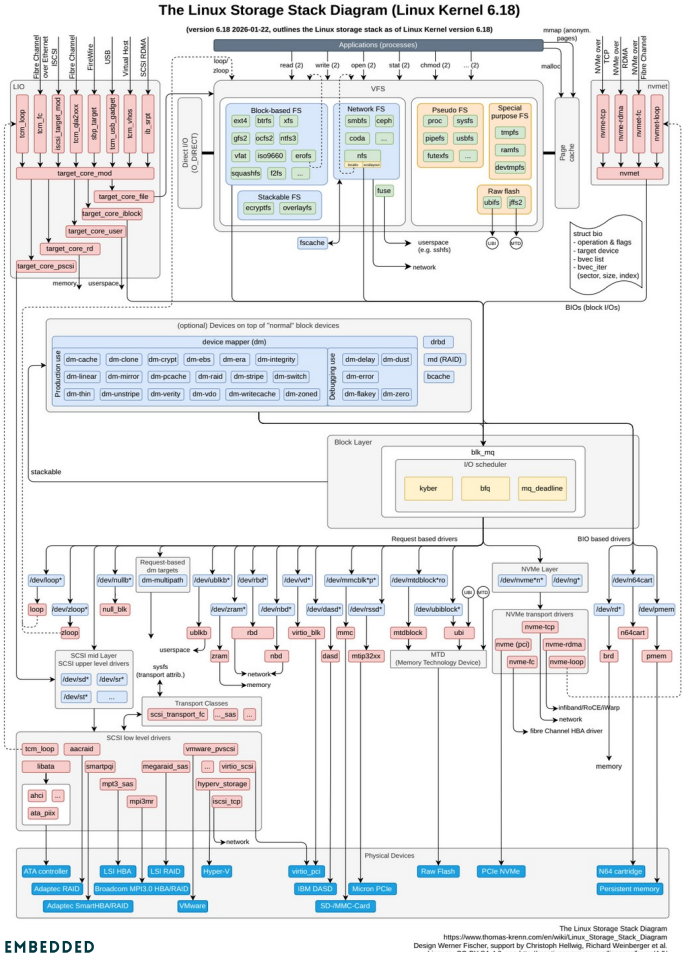
Pic: © Felix Fischer / CC-BY-SA-4.0

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
EUROPE
Embedded Linux Conference

Linux Storage Stack overview

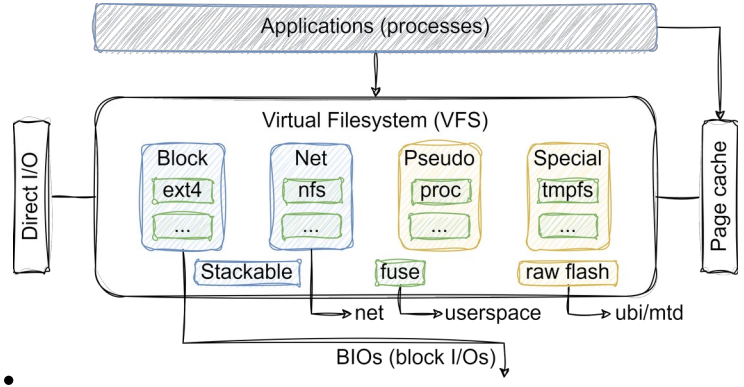



Applications (processes)
lio
Direct I/O
Virtual Filesystem (VFS)
Block
Net
Pseudo
Special
net
Page cache
nvmet
BIOs (block I/Os)
remapper
device mapper (dm)
drbd
md
bcache
BIOs (block I/Os)
Block Layer / blk-mq
Per Core SW Queues
HW Dispatch Queues
Requests (each one or more BIOs)
request-based drivers
bio-based drv.
Physical devices (HDDs, SSDs, NAND, HW-RAID, ...
N64 cartridge

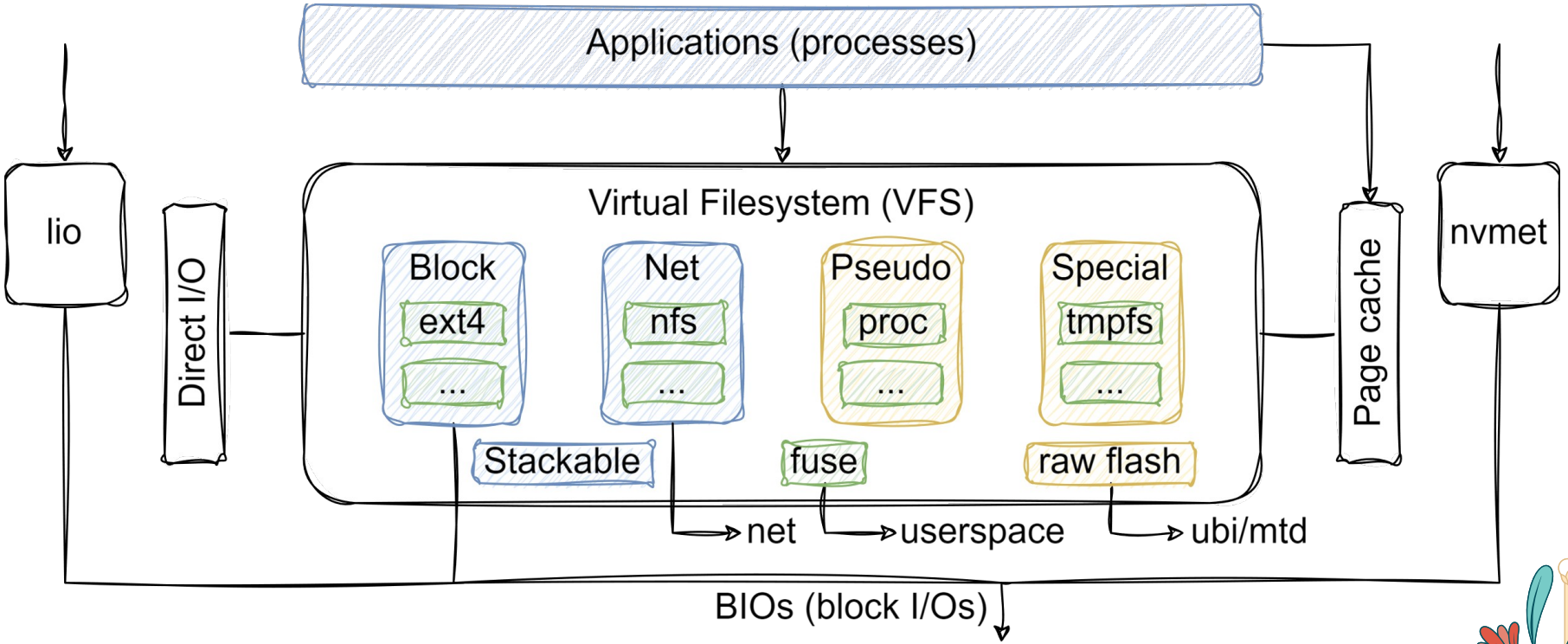


The Linux Storage Stack Diagram
https://www.thomas-krenn.com/en/wiki/Linux_Storage_Stack_Diagram
Design Werner Fischer, support by Christoph Hellwig, Richard Weinberger et al.

Virtual File System (VFS)

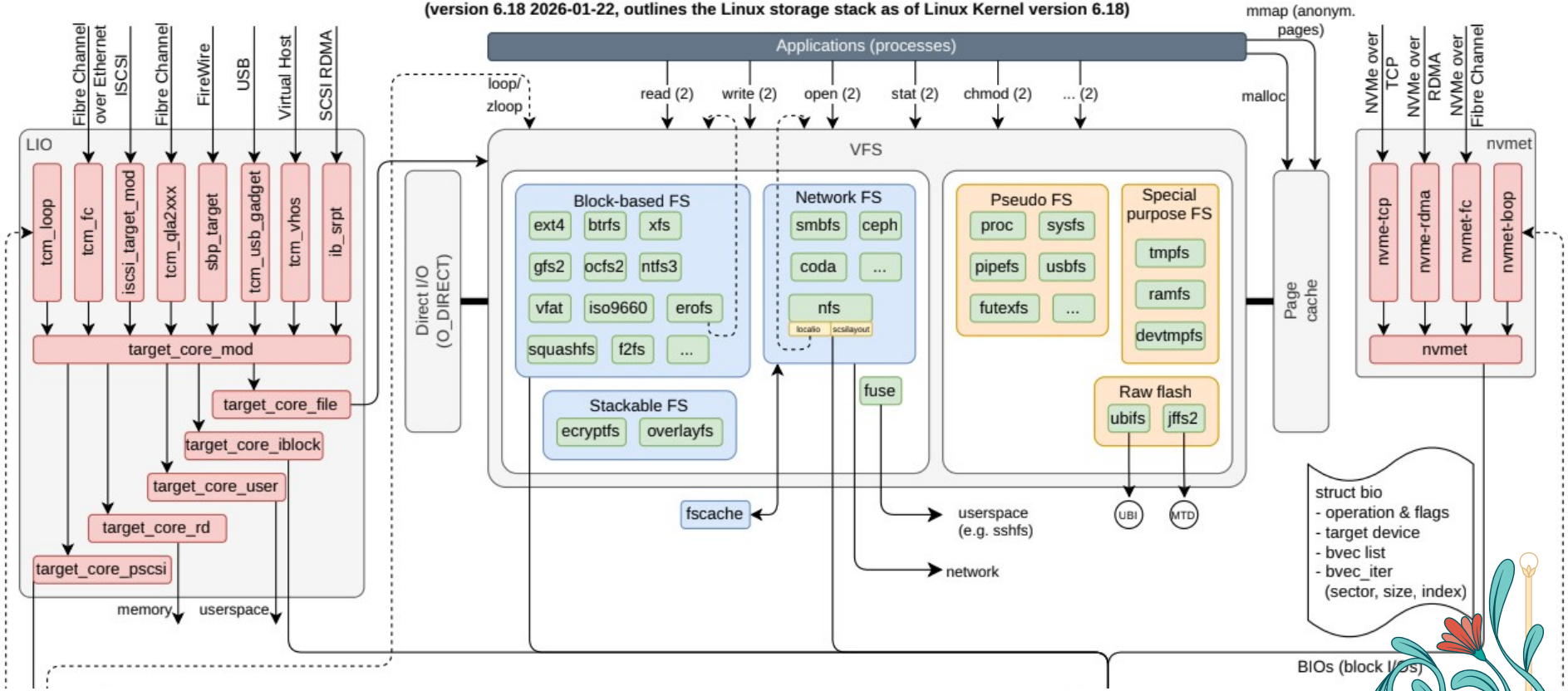


- introduced with System V
- generic interface for applications
- operations like read(2), write(2), open(2) work identical, independent of the filesystem being used


Applications (processes)
lio
Direct I/O
Virtual Filesystem (VFS)
Block
ext4
...
Net
nfs
...
Pseudo
proc
...
Special
tmpfs
...
Page cache
nvmet
Stackable
fuse
raw flash
net
userspace
ubi/mtd
BIOs (block I/Os)

The Linux Storage Stack Diagram (Linux Kernel 6.18)

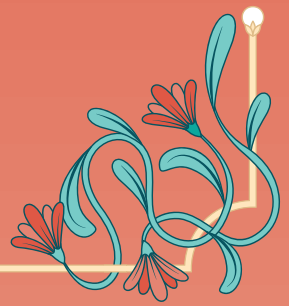
(version 6.18 2026-01-22, outlines the Linux storage stack as of Linux Kernel version 6.18)



THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
EUROPE

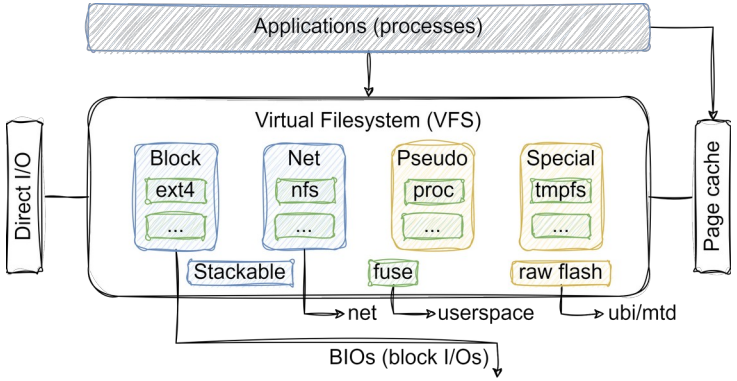
Embedded Linux Conference

Linux Page Cache: Making I/O Fast

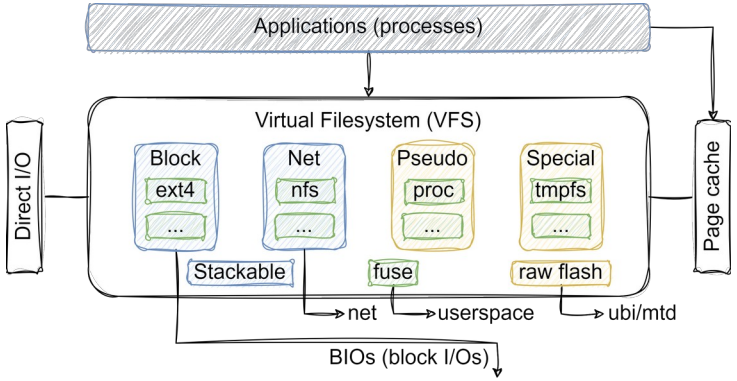


Linux Page Cache

- Linux uses unused RAM to cache **file data** read from / written to storage.

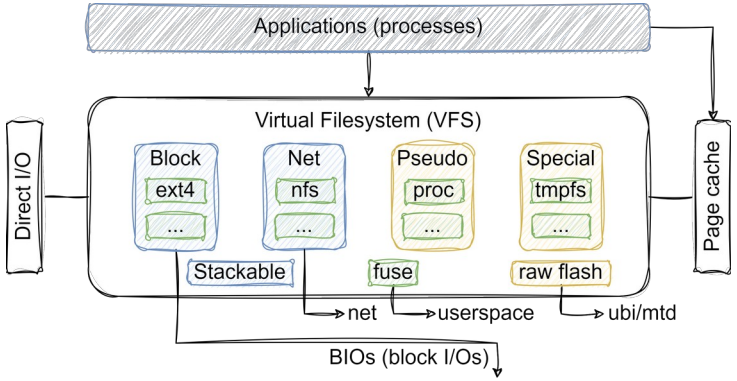


Linux Page Cache



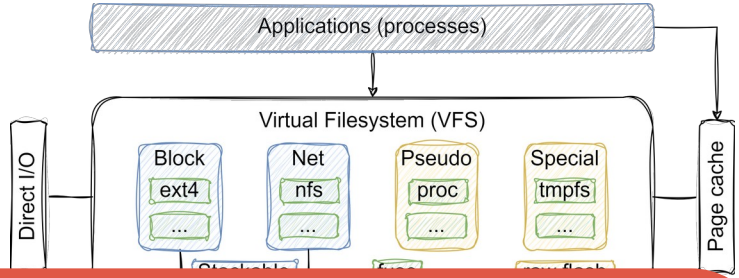
- Linux uses unused RAM to cache **file data** read from / written to storage.
- When an application reads a file:
  - **Cache hit**: data is already in RAM → serve directly.
  - **Cache miss**: read from disk → place the data in page cache.

Linux Page Cache



- Linux uses unused RAM to cache **file data** read from / written to storage.
- When an application reads a file:
  - **Cache hit**: data is already in RAM → serve directly.
  - **Cache miss**: read from disk → place the data in page cache.
- Writes normally go to RAM first and make the corresponding pages **dirty**.
  - Dirty pages are later written back to storage by kernel's **writeback** mechanism.
  - sync, fsync(), memory pressure, or background/threshold-based writeback can cause dirty data to be written.

Linux Page Cache



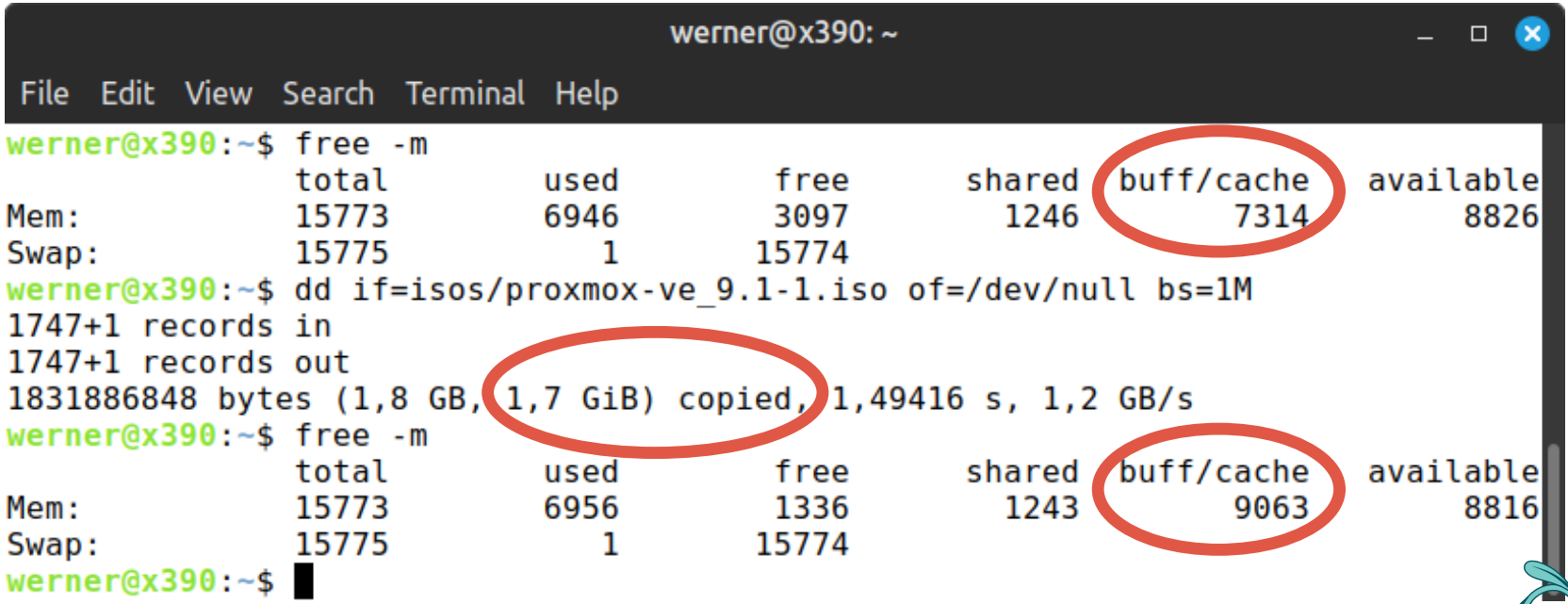
- Linux uses unused RAM to cache **file data** rea
- When an ap
  - **Cache**
  - **Cache**
- s no
  ty pag
  , fsync(), memory pressure, or background/threshold-based writeback
  ause dirty data to be written.

Key idea:

RAM acts as a cache and write buffer between apps and relatively slow storage



Page Cache


werner@x390: ~
File Edit View Search Terminal Help
werner@x390:~$ free -m
total used free shared buff/cache available
Mem: 15773 6946 3097 1246 7314 8826
Swap: 15775 1 15774
werner@x390:~$ dd if=isos/proxmox-ve_9.1-1.iso of=/dev/null bs=1M
1747+1 records in
1747+1 records out
1831886848 bytes (1,8 GB, 1,7 GiB) copied, 1,49416 s, 1,2 GB/s
werner@x390:~$ free -m
total used free shared buff/cache available
Mem: 15773 6956 1336 1243 9063 8816
Swap: 15775 1 15774
werner@x390:~$

Tuneables in /proc/sys/vm/

Parameter	Meaning
vm.dirty_background_ratio	% of available memory at which the kernel **starts background writeback**.
vm.dirty_ratio	% of available memory at which processes generating dirty data are forced to **participate in writeback** / are throttled, preventing dirty memory from growing without bound.
vm.dirty_expire_centisecs (Unit = 1/100 sec.)	How long a dirty page can remain dirty before it becomes old enough to be **considered for writeback**.
vm.dirty_writeback_centisecs (Unit = 1/100 sec.)	How **frequently** the kernel's periodic writeback mechanism (kernel flusher threads) wakes up to inspect/write dirty pages.

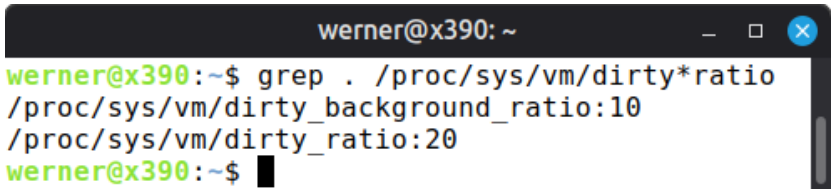
Tuneables

```
File  Edit  View  Search  Terminal  Help
werner@x390:~$ grep . /proc/sys/vm/dirty_*
/proc/sys/vm/dirty_background_bytes:0
/proc/sys/vm/dirty_background_ratio:10
/proc/sys/vm/dirty_bytes:0
/proc/sys/vm/dirty_expire_centisecs:1500
/proc/sys/vm/dirty_ratio:20
/proc/sys/vm/dirty_writeback_centisecs:1500
werner@x390:~$ uname -a
Linux x390 6.1.0-43-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.162-1 (2026-02-08) x
86_64 GNU/Linux
werner@x390:~$
```

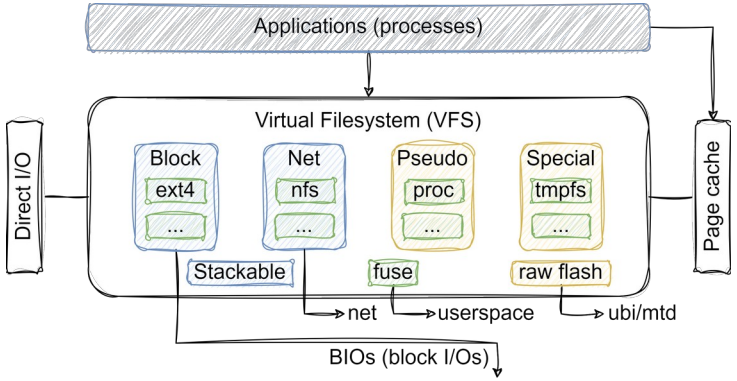
```
werner@t14:~$ grep . /proc/sys/vm/dirty_*
/proc/sys/vm/dirty_background_bytes:0
/proc/sys/vm/dirty_background_ratio:10
/proc/sys/vm/dirty_bytes:0
/proc/sys/vm/dirty_expire_centisecs:3000
/proc/sys/vm/dirty_ratio:20
/proc/sys/vm/dirty_writeback_centisecs:500
werner@t14:~$ uname -a
Linux t14 6.12.73+deb13-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.12.73-1 (2026-02-1
7) x86_64 GNU/Linux
werner@t14:~$
```

```
root@pve:~# grep . /proc/sys/vm/dirty_*
/proc/sys/vm/dirty_background_bytes:0
/proc/sys/vm/dirty_background_ratio:10
/proc/sys/vm/dirty_bytes:0
/proc/sys/vm/dirty_expire_centisecs:3000
/proc/sys/vm/dirty_ratio:20
/proc/sys/vm/dirty_writeback_centisecs:500
root@pve:~# uname -a
Linux pve 6.17.2-1-pve #1 SMP PREEMPT_DYNAMIC PMX 6.17.2-1 (2025-10-21T11:55Z) x
86_64 GNU/Linux
root@pve:~#
```

Linux Page Cache ratios







- Below 10% dirty → no threshold-triggered background writeback.
- Above 10% → background flusher starts writing.
- Above 20% → applications producing dirty data can be forced to wait.

Important: these are ratios of **available memory**,
not simply percentages of total physical RAM!

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
EUROPE
Embedded Linux Conference

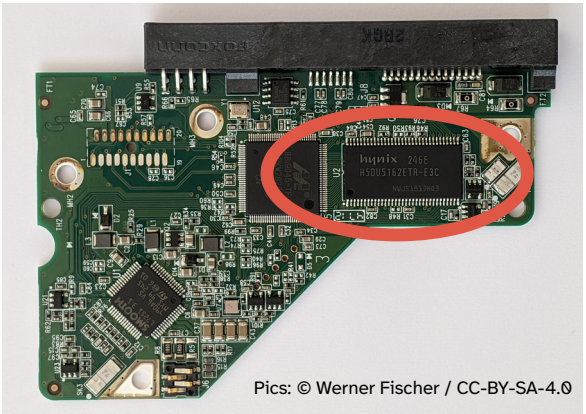
Caches: HDDs, SSDs, Storages



Example: HDD

- 64 MB cache
- data in cache gets lost in case of power outage





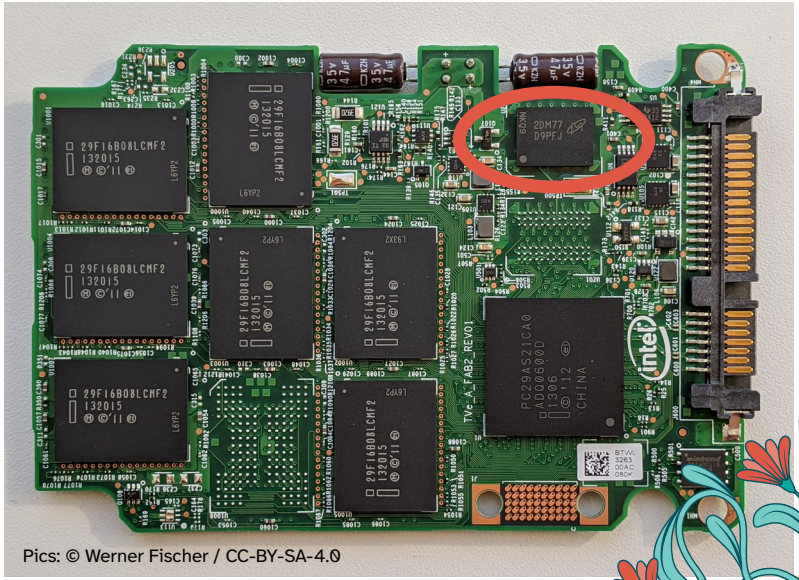
Pics: © Werner Fischer / CC-BY-SA-4.0



Example: Datacenter SSD

- cache is protected by **power-loss protection**
- in the event of a **power failure**, the **cache contents is written** to non-volatile flash memory (using energy from the capacitors)
- contents of the cache is not lost in the event of a power failure!





Pics: © Werner Fischer / CC-BY-SA-4.0

Example: some Datacenter HDDs

- e.g. Toshiba MG10-D
- Toshiba Persistent Write Cache (PWC) Technology for Data-Loss Protection in Sudden Power-Loss Events



Pics: © Werner Fischer / CC-BY-SA-4.0

PWC (Persistent Write Cache) with PLP (Power Loss Protection): PWC with PLP is a function to handle the write data that the drive reports "Normal completion" to the host but not being stored to hard disk media yet.

The write data may be written to the commanded LBA on the hard disk media. The un-written data to hard disk media is stored to **Flash memory** using **back up power by PLP** when the power supply to the drive suddenly is shut down. And, after PLP operation, it may be required more time to start up the drive than in case of normal shutdown.

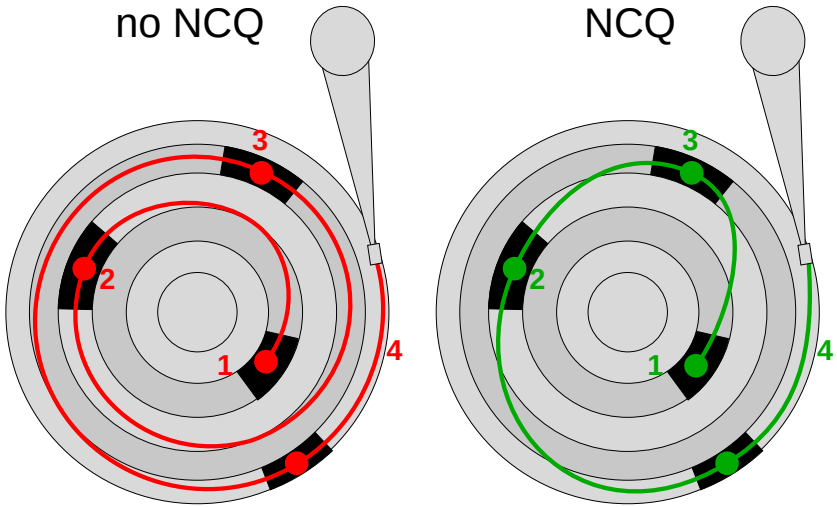
1) PLP does not secure data in the mode of all the power shutdowns. When power supplies other than recommended procedure are intercepted, data might be lost.

2) In the power shutdown before it reports on the Write completion, data not anticipated might be lost.

Source: https://toshiba.semicon-storage.com/content/dam/toshiba-ss-v3/master/en/storage/product/data-center-enterprise/eHDD-MG04ACAxxx-Product-Overview.pdf

Write Ordering

- HDDs/SSDs can reorder data (1→2→3→4 as 1→3→2→4 on disk) → Native Command Queuing (NCQ)
- However, file system journals rely on the correct order (1. Data, 2. Journal, 3. Mark commit as complete)

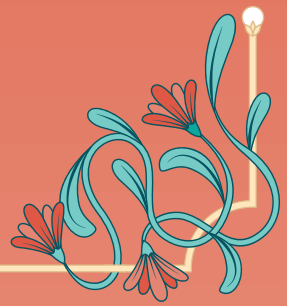


Source: https://commons.wikimedia.org/wiki/File:NCQ.svg / CC-BY-SA 3.0

File System Write Barriers protect

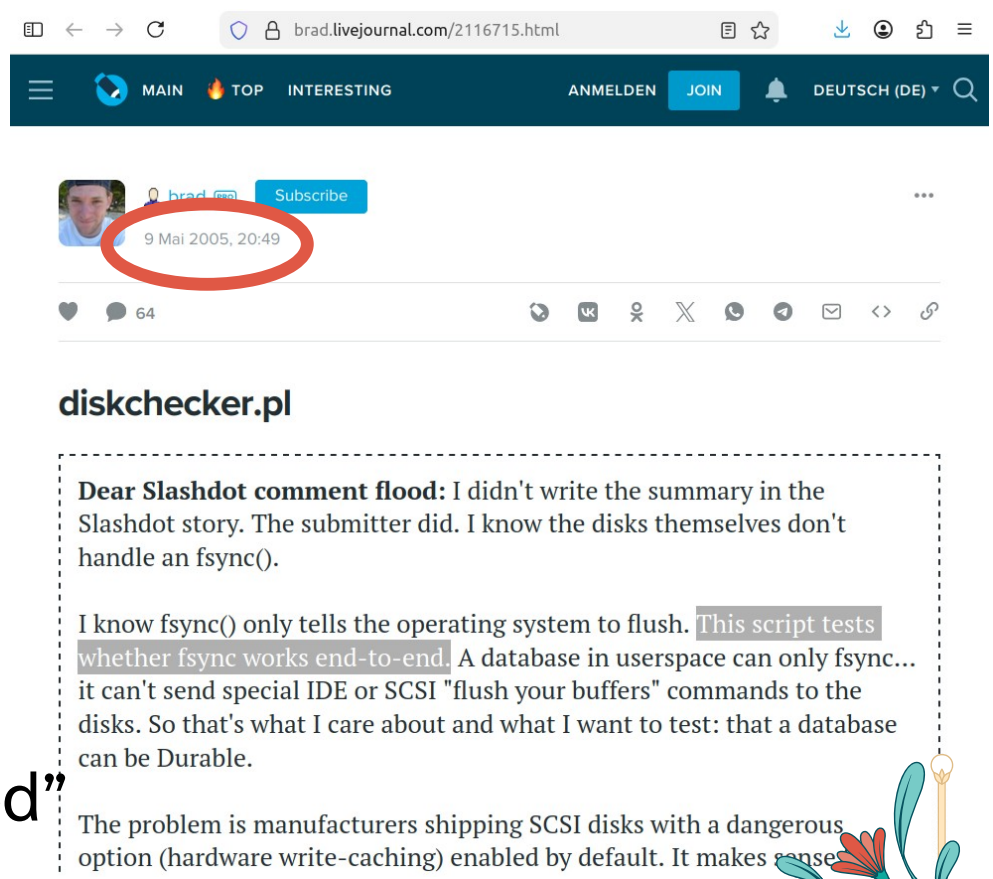
- Write barriers ensure that all data preceding the barrier is written to permanent storage before any further data follows.
- e.g., ext4/JBD2 implements journal write barriers using the block layer's persistence primitives — notably **FLUSH** (REQ_PREFLUSH) and **FUA** (REQ_FUA).

ext4 barrier=0 for higher performance?
→ not a good idea



diskchecker.pl

- Perl script
  - validates fsync() function
- 2 systems for tesing
  - “Tester machine”
  - “Machine being tested”



Tester machine (machine that won't crash):

```
$ diskchecker.pl -l
```

Machine being tested (machine you're going to pull the power cable on)

```
$ diskchecker.pl -s TESTERMACHINE create test_file 500
```

Now, pull the power cable on the machine being tested. Don't turn it off nicely. Don't just control-C the program. Wait a couple seconds and plug your testee machine back in and reboot it. When it's back up, do:

```
$ diskchecker.pl -s TESTERMACHINE verify test_file
```

If the server process is still running, the machine you just killed will connect to the server and information about what's supposed to be where. The client will then verify it and produce an error report.

What you **should** see is:

```
Total errors: 0.
```

www.thomas-krenn.com/de/wiki/Ext4_Write_Barriers

THOMAS KRENN | Thomas-Krenn-Wiki durchsuchen | Onlineshop | DE

Ext4 Dateisystem als Beispiel

Das Ext4 Dateisystem verwendet seit Mai 2008 standardmäßig Write Barriers.[1] Dadurch wird sichergestellt, dass auch bei Datenträgern mit flüchtigen Caches - z.B. bei Festplatten mit RAM-Caches - selbst bei einem plötzlichen Stromausfall keine Daten verloren gehen, dis mittels fsync() geschrieben wurden.

Die Aktivierung von Schreibbarrieren kann bei einigen Anwendungen zu erheblichen Leistungseinbußen führen. Insbesondere Anwendungen, die fsync() intensiv nutzen oder viele kleine Dateien erstellen und löschen, laufen mitunter deutlich langsamer.[2] Wenn RAID-Controller mit battery-backed Cache oder bespielsweise SSDs mit SSD Power Loss Protection verwendet werden, kann das Deaktivieren von write barriers die Performance steigern, ohne das Risiko eines Datenverlustes zu erhöhen.[3] Das gilt jedoch nur dann, wenn tatsächlich eine funktionierende Power-Loss-Protection vorhanden ist.

Testfall	HDD Western Digigital WD5003ABYX	Intel SSD mit PLP	Samsung SSD ohne PLP
barriers=1 (aktiviert)	**kein Datenverlust**	**kein Datenverlust**	**kein Datenverlust**
barriers=0 (deaktiviert)	**Datenverlust**	**kein Datenverlust**	kein Datenverlust (Achtung: Datenverlust dennoch nicht ausgeschlossen!)

Ext4 Dateisystemtests mit diskchecker.pl

Die folgendes Tests wurden alle mit dem diskchecker.pl Perl Skript durchgeführt. Mit dem Testscript wird mithilfe von zwei Rechnern geprüft, ob Daten bei einem Stromausfall verloren gehen können. Dazu wird das Skript auf zwei Rechnern ausgeführt:

1. auf einem Protokollierungs-Rechner: dort läuft das Skript im Servermodus.
2. auf dem zu testenden System: dort wird nachdem das Skript gestartet wurde, die Stromzuvor unterbrochen.

Danach wird das zu testende System erneut angesteckt und gestartet. Das Skript wird nun erneut zur Verifizierung ausgeführt. Weitere Hintergrundinformationen zu diesem Skript gibt es hier: http://brad.livejournal.com/2116715.html

Test mit WD 500 GByte HDD

In diesem Test wird eine Western Digital WD5003ABYX 500GB SATA HDD mit 64MB Cache verwendet.



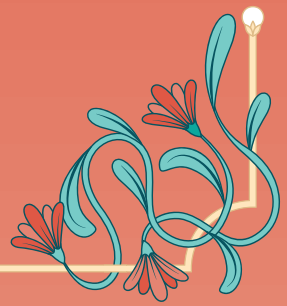
barrier=1 (default)

```
root@pve:~# mount | grep sdc
/dev/sdc1 on /mnt/sdc-hdd type ext4 (rw,relatime)
root@pve:~# uname -a
Linux pve 6.17.2-1-pve #1 SMP PREEMPT_DYNAMIC PMX 6.17.2-1 (2025-10-21T11:55Z)
x86_64 GNU/Linux
root@pve:/mnt/sdc-hdd# ./diskchecker.pl -s 172.16.0.112 create test-hdd-ext4-
default 1000
  diskchecker: running 1 sec, 0.02% coverage of 1000 MB (11 writes; 11/s)
  diskchecker: running 2 sec, 0.05% coverage of 1000 MB (35 writes; 17/s)
```

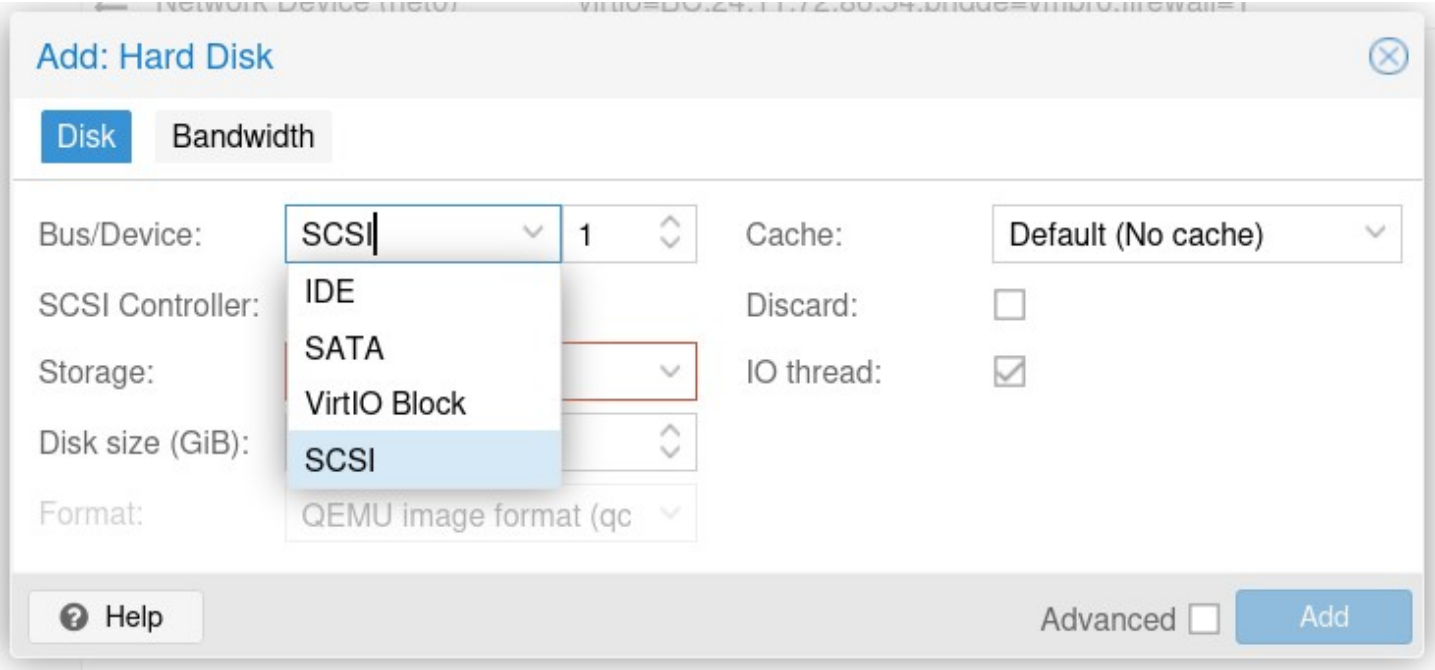
```
[...]
root@pve:~# mount -t ext4 -o barrier=0 /dev/sdc1 /mnt/sdc-hdd
root@pve:~# dmesg | tail -n 4
[   38.535100] vmbr0: port 1(nic1) entered forwarding state
[  324.430790] EXT4-fs (sdc1): barriers disabled
[  327.063895] EXT4-fs (sdc1): recovery complete
[  327.064076] EXT4-fs (sdc1): mounted filesystem 8ac821f6-c372-415c-addd-855c3161ded9 r/w with ordered data mode. Quota mode:
none.
root@pve:~# cd /mnt/sdc-hdd
root@pve:/mnt/sdc-hdd# ./diskchecker.pl -s 172.16.0.112 verify test-hdd-ext4-nobarrier
 verifying: 0.02%
  Error at page 19, 2 seconds before end.
  Error at page 303, 1 seconds before end.
  Error at page 339, 0 seconds before end.
  Error at page 511, 1 seconds before end.
  Error at page 544, 0 seconds before end.
  Error at page 681, 0 seconds before end.
  Error at page 708, 2 seconds before end.
  Error at page 1108, 1 seconds before end.
  Error at page 1337, 0 seconds before end.
  Error at page 1388, 0 seconds before end.
 verifying: 2.22%
  Error at page 1487, 0 seconds before end.
  Error at page 1492, 0 seconds before end.
[...]
  Error at page 63758, 0 seconds before end.
  Error at page 63990, 1 seconds before end.
  verifying: 100.0
Total errors: 545
Histogram of seconds before end:
    0  178
    1  187
    2  142
    3    1
    4    1
```

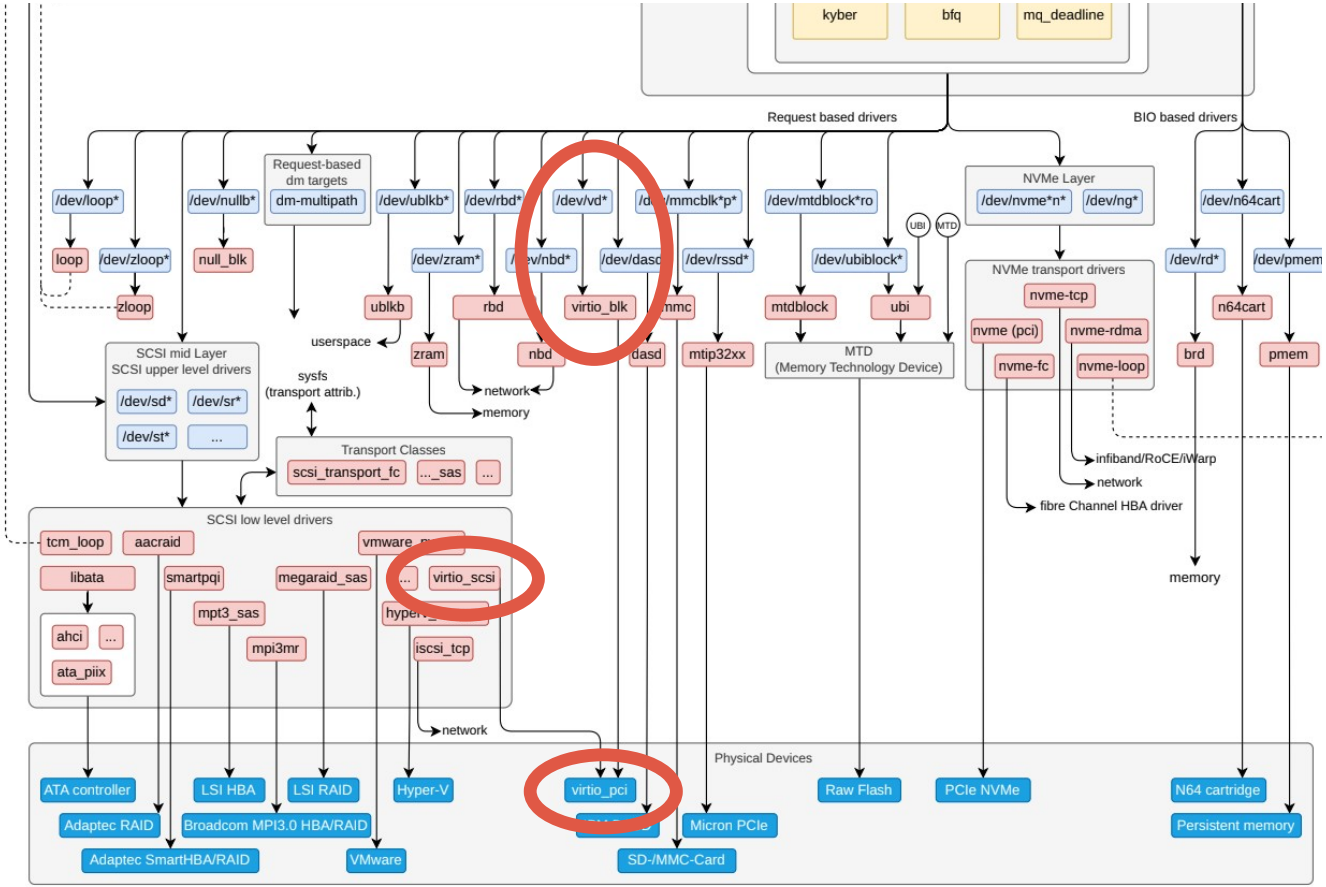
THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
EUROPE
Embedded Linux Conference

Caches: KVM / QEMU / Proxmox VE



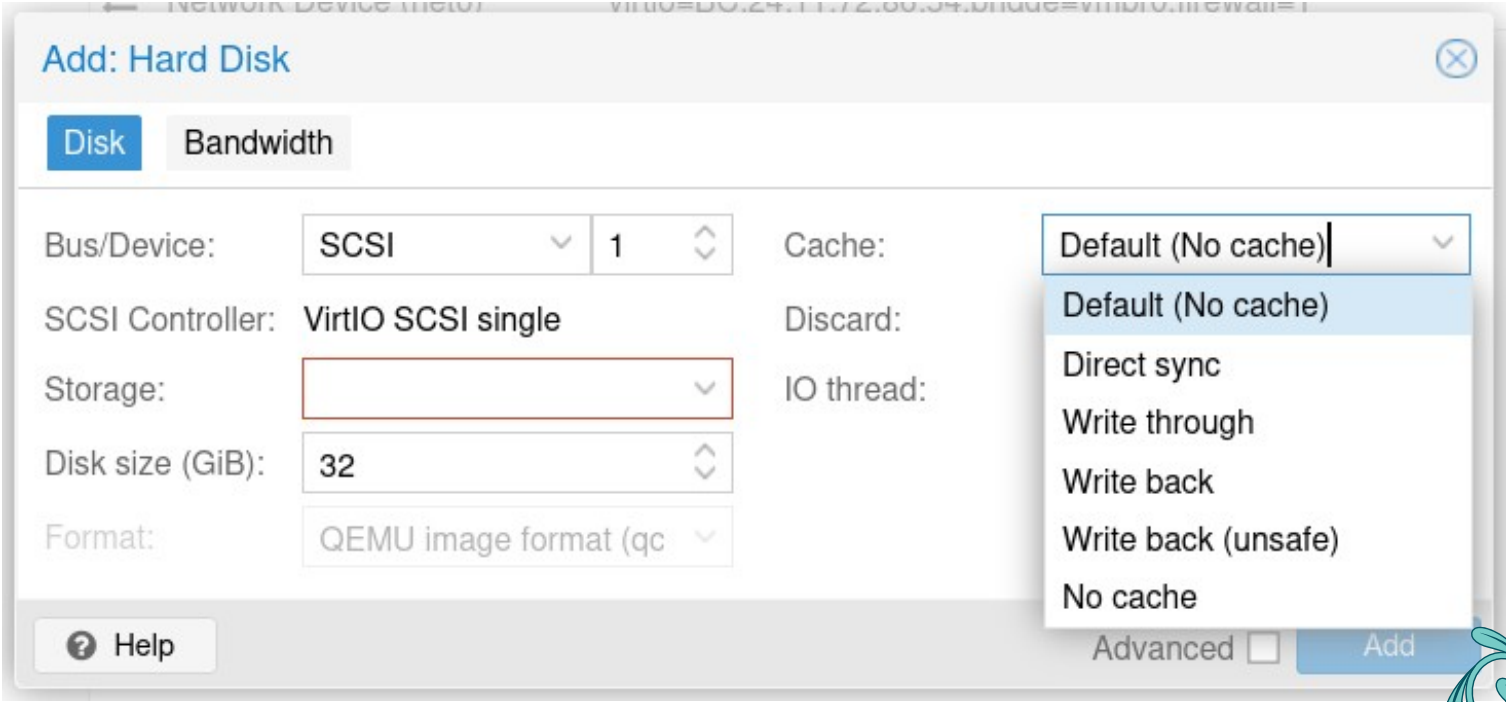
Proxmox VE Add Hard Disk





The Linux Storage Stack Diagram

Proxmox VE Add Hard Disk



QEMU disk caching: three independent concepts

Setting	Question it answers
-device cache.writeback	Does the **virtual device expose a write-back cache to the guest?**
-blockdev cache.direct	Does QEMU **use the host page cache?**
-blockdev cache.no-flush	Does QEMU **honor guest flush requests?**

QEMU legacy cache setting

cache		-device option	-blockdev options	
Name QEMU	Proxmox VE GUI	cache.writeback	cache.direct	cache.no-flush
writeback	Write back	✔	-	-
none	No cache	✔	✔	-
writethrough	Write through	-	-	-
directsync	Direct sync	-	✔	-
unsafe	Write back (unsafe)	✔	-	✔

1) -device cache.writeback

- **virtual disk cache** in the guest (just like with a non-virtualized HDD/SSD)
- guest OS must properly flush this virtual and volatile disk cache — just as it would with non-virtualized HDDs using **write barriers**
- The host uses the page cache for the virtual disk cache (unless disabled via `cache.direct`) as well as other caches, such as HDD/SSD caches

1) -device cache.writeback

Disk (Proxmox VE GUI)	query cache via	result with cache.writeback=		note
		on	off	
IDE	**hdparm -I /dev/sdc | grep "Write cache"**	* Write cache	Write cache (no „*“)	hdparm executes an ATA IDENTIFY DEVICE command to query the setting.
SATA				
VirtIO Block	**cat /sys/block/vda/queue/write_cache**	write back	write through	The VirtIO block driver makes the information available via sysfs.
SCSI	**sdparm --get=WCE /dev/sda**	WCE = 1	WCE = 0	sdparm executes a SCSI MODE SENSE command to query the setting. WCE stands for “Writeback Cache Enable.”

2) -blockdev cache.direct

- cache.direct=on bypasses the host page cache
- the hypervisor accesses the disk image file or block device using **O_DIRECT** semantics

3) -blockdev cache.no-flush

- **high risk of data loss!**
- cache.no-flush instructs QEMU never to flush data to permanent storage
- in the event of a power outage, the image will likely become unusable
- can be used temporarily for installing a guest OS, but be sure to disable it afterward!

Test results diskchecker.pl

Cache setting	diskchecker.pl result
Write Back	Total errors: 0
No cache	Total errors: 0
Write through	Total errors: 0
Direct Sync	Total errors: 0
Write back (unsafe)	**Total errors: 2786**

Write back (unsafe)

```
tk@debian13-1:~$ ./diskchecker.pl -s 172.16.0.112 create test_file_unsafe 1000
  diskchecker: running 1 sec, 0.45% coverage of 1000 MB (286 writes; 286/s)
  diskchecker: running 2 sec, 1.22% coverage of 1000 MB (782 writes; 391/s)
  diskchecker: running 3 sec, 2.06% coverage of 1000 MB (1328 writes; 442/s)
  diskchecker: running 4 sec, 2.81% coverage of 1000 MB (1820 writes; 455/s)
  diskchecker: running 5 sec, 3.55% coverage of 1000 MB (2306 writes; 461/s)
  diskchecker: running 6 sec, 4.25% coverage of 1000 MB (2778 writes; 463/s)
  diskchecker: running 7 sec, 5.04% coverage of 1000 MB (3302 writes; 471/s)
  diskchecker: running 8 sec, 5.85% coverage of 1000 MB (3863 writes; 482/s)
  diskchecker: running 9 sec, 6.73% coverage of 1000 MB (4466 writes; 496/s)
  diskchecker: running 10 sec, 7.50% coverage of 1000 MB (4984 writes; 498/s)
[...]
tk@debian13-1:~$ ./diskchecker.pl -s 172.16.0.112 verify test_file_unsafe
 verifying: 0.02%
  Error at page 26, 4 seconds before end.
  Error at page 34, 6 seconds before end.
  Error at page 49, 3 seconds before end.
  Error at page 123, 0 seconds before end.
  Error at page 166, 0 seconds before end.
  Error at page 167, 6 seconds before end.
  Error at page 169, 1 seconds before end.
  Error at page 199, 1 seconds before end.
  Error at page 200, 2 seconds before end.
  Error at page 214, 4 seconds before end.
[...]
 verifying: 100.00%
Total errors: 2786
Histogram of seconds before end:
     0  420
     1  478
     2  339
     3  285
     4  303
     5  351
     6  354
     7  256
tk@debian13-1:~$
```

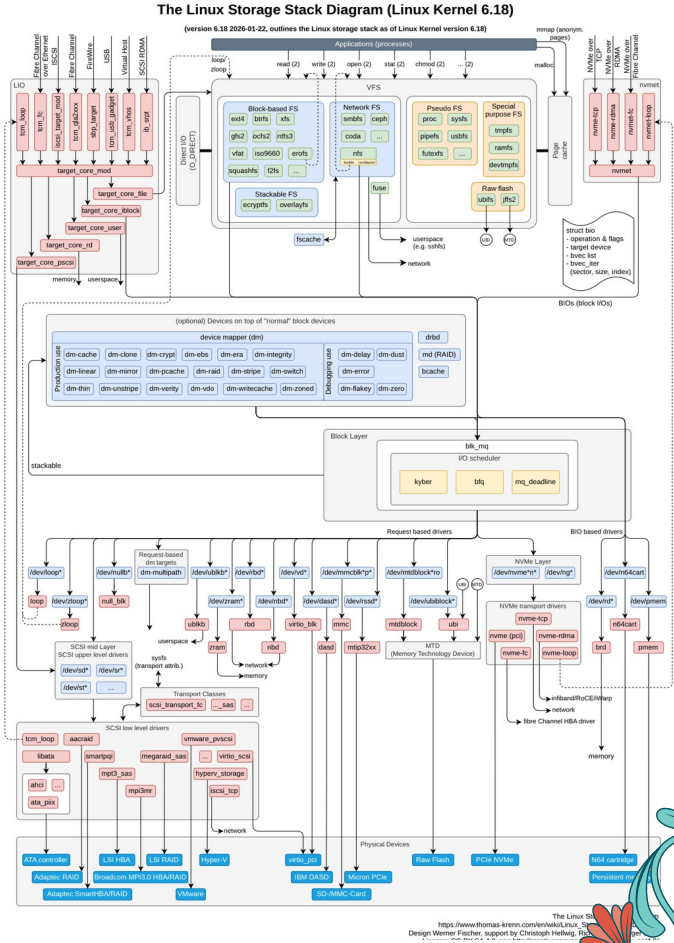
cache		-device option	-blockdev options		Ensuring Data Integrity During a Power Outage	write perf.
Namo QEMU	Proxmox VE GUI	cache. writeback	cache. direct	cache. no-flush		
writeback	Write back	✔	-	-	High when the guest OS performs cache flushes (write barriers)	high
none	No cache	✔	✔ O_DIRECT	-		
writethrough	Write through	- O_DSYNC	-	-	Very high, because QEMU calls fsync() after every write operation	lower
directsync	Direct sync	- O_DSYNC	✔ O_DIRECT	-		
unsafe	Write back (unsafe)	✔	-	✔	None—high risk!	highest

Lessons Learned

- Proxmox' default setting "No cache" (none) makes a lot of sense.
- During Guest OS installation, you can use "Write back (unsafe)" → BUT be sure to set it back to "No cache" afterward!
- Perl code still works even after >20 years :-) diskchecker.pl is still valuable today!

Thanks to Christoph Hellwig, Richard Weinberger, Hannes Reinecke, and many other Linux developers for their valuable suggestions and contributions.

Without their help, I still wouldn't have a clue how the Linux storage stack works.



THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
EMBEDDED LINUX CONFERENCE
EUROPE

THE LINUX FOUNDATION



OPEN SOURCE SUMMIT



EMBEDDED LINUX CONFERENCE

EUROPE

