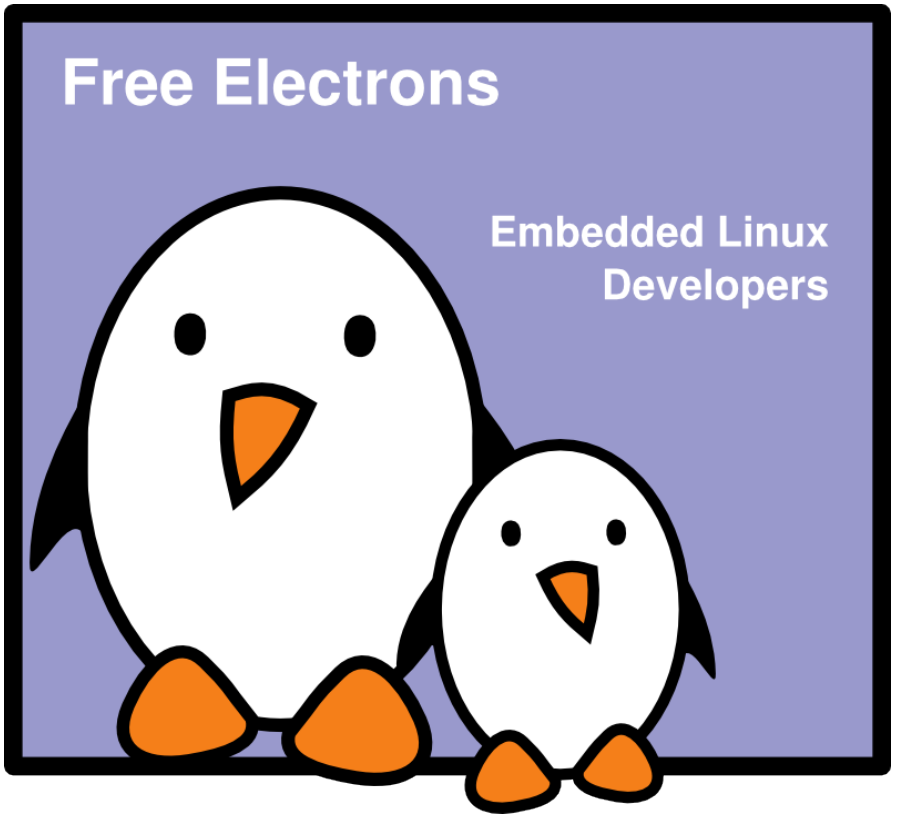
Flash filesystem benchmarks

Michael Opdenacker
Free Electrons

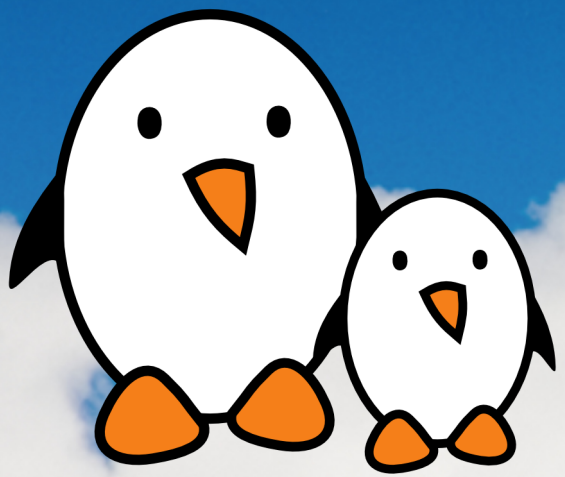


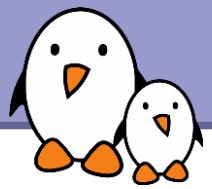
Free Electrons

Free embedded Linux and kernel materials
http://free-electrons.com/docs

Linux BSPs, device drivers and plumbing

Buildroot
http://buildroot.org

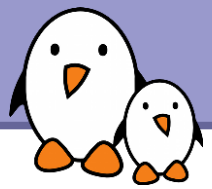




Questions

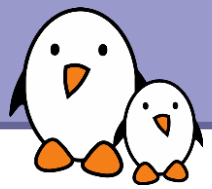
Who uses?

- jffs2
- ubifs
- yaffs2
- logfs
- nftl?



Introduction

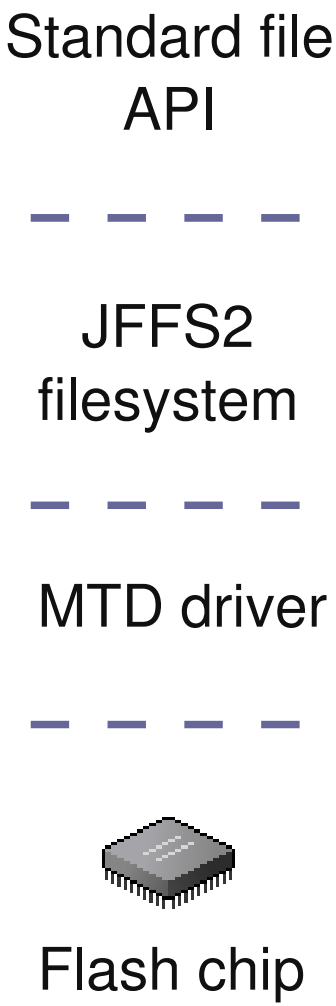
- Work started in 2008 at ELC-E
  Showed benchmarks on jffs2, yaffs2 and ubifs
- What has happened during the last 2 years
  - Project continued with funding from the CE Linux Forum
  - Automation scripts now supporting multiple boards
  - New `armel` root filesystem for the tests (replaces Buildroot).
    Easier access to MTD and filesystem utilities.
    Udev simplifies the use of UBIFS.
  - Now measuring driver initialization time through loading external modules (instead of static drivers - boot time was not measured).
  - LogFS mainlined in 2.6.34

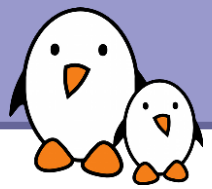


jffs2

http://www.linux-mtd.infradead.org/doc/jffs2.html

- Today's standard filesystem for MTD flash
- Nice features: on the fly compression (saves storage space and reduces I/O), power down reliable, wear-leveling and ECC.
- Drawbacks: doesn't scale well
  - Mount time depending on filesystem size: the kernel must scan the whole filesystem at mount time, to read which block belongs to each file.
  - Need to use the `CONFIG_JFFS2_SUMMARY` kernel option to store such information in flash. This dramatically reduces mount time (from 16 s to 0.8s for a 128 MB partition).

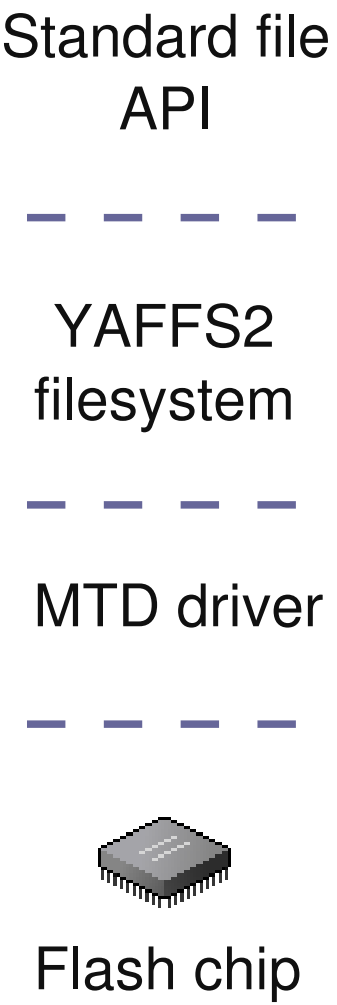


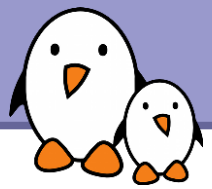


yaffs2

http://www.yaffs.net/

- Mainly supports NAND flash
- No compression
- Wear leveling, ECC, power failure resistant
- Fast boot time
- Code available in a separate `git` tree
  Should be included in the mainline kernel soon.

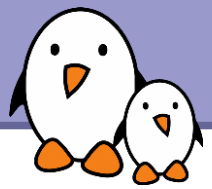




UBI (1)

Unsorted Block Images

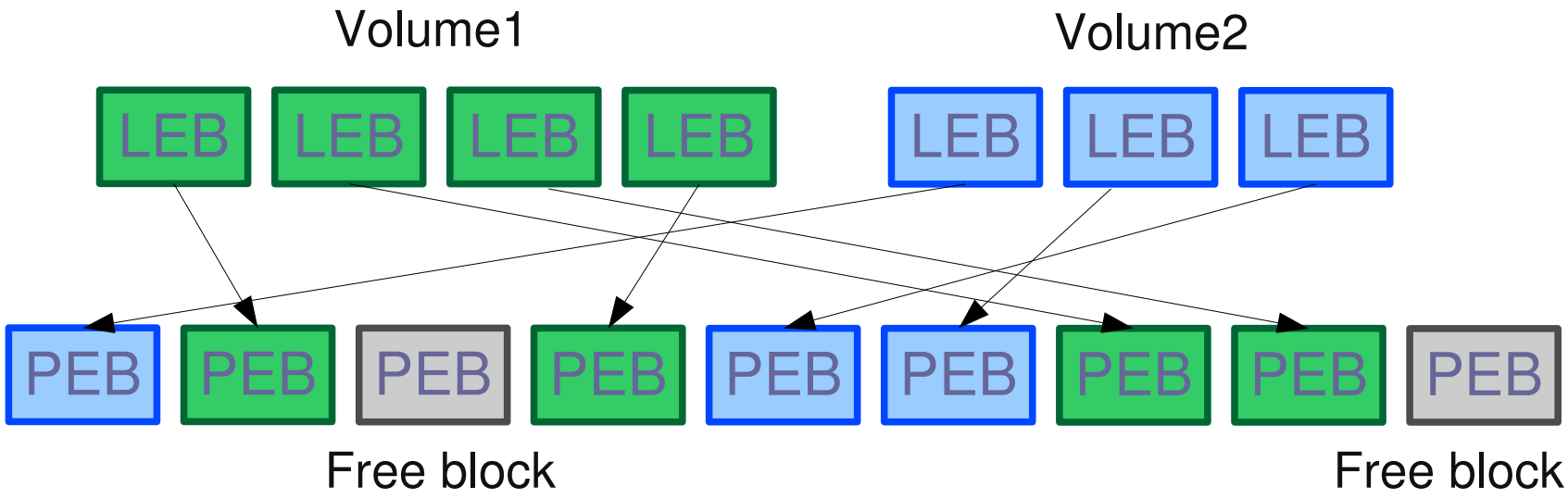
- http://www.linux-mtd.infradead.org/doc/ubi.html
- Volume management system on top of MTD devices.
- Allows to create multiple logical volumes and spread writes across all physical blocks.
- Takes care of managing the erase blocks and wear leveling. Makes filesystem easier to implement.

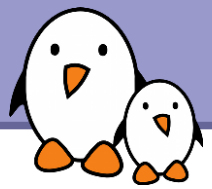


UBI (2)

UBI
Logical
Erase Blocks

MTD
Physical
Erase Blocks

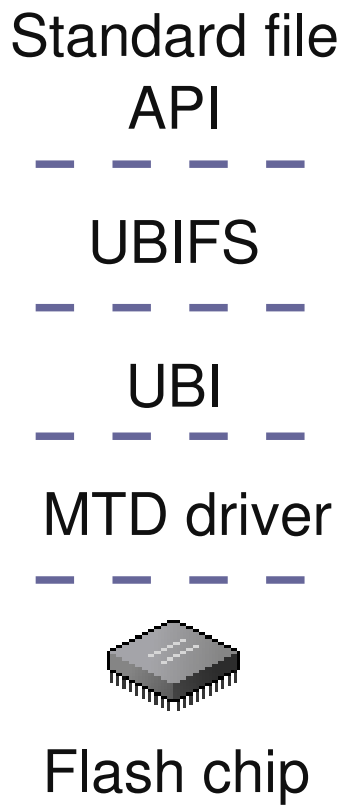


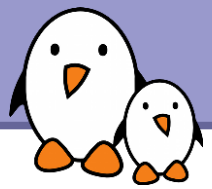


UBIFS

http://www.linux-mtd.infradead.org/doc/ubifs.html

- The next generation of the jffs2 filesystem, from the same linux-mtd developers.
- Available in Linux 2.6.27
- Works on top of UBI volumes
- Has a noticeable metadata overhead on very small partitions (4M, 8M)

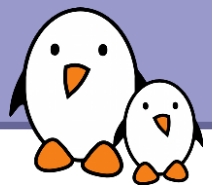




LogFS

http://en.wikipedia.org/wiki/LogFS

- New comer in mainline (integrated in 2.6.34)
- Still experimental (at least in 2.6.36)
- Designed to be very fast at mount time
  (even faster than UBIFS): O(1) mount time
- Supposed to consume less RAM than JFFS2
- Ability to run on block devices but with poor performance

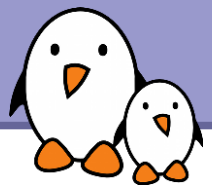


Logfs in action

```
kernel BUG at fs/logfs/segment.c:858!
Unable to handle kernel NULL pointer dereference at virtual address
00000000
pgd = ced00000
[00000000] *pgd=8edbd031, *pte=00000000, *ppte=00000000
Internal error: Oops: 817 [#1]
last sysfs file: /sys/devices/virtual/vc/vcsa6/uevent
Modules linked in: logfs zlib_deflate
CPU: 0    Tainted: G        W    (2.6.36 #2)
PC is at __bug+0x1c/0x28
LR is at __bug+0x18/0x28
pc : [<c002b428>]    lr : [<c002b424>]    psr: 20000013
sp : cfb7fe38  ip : 00000928  fp : 00000080
r10: c0580b20  r9 : 00000009  r8 : ffffffff
r7 : cfb7fea4  r6 : 00000080  r5 : cf50cd58  r4 : c0580b20
r3 : 00000000  r2 : cfb7fe2c  r1 : c02fd4b7  r0 : 0000002c
Flags: nzCv  IRQs on  FIQs on  Mode SVC_32  ISA ARM  Segment user
Control: 10c5387d  Table: 8ed00019  DAC: 00000015
Process umount (pid: 868, stack limit = 0xcfb7e2e8)
Stack: (0xcfb7fe38 to 0xcfb80000)
fe20:
bf0122dc bf0122ec
```

With Linux 2.6.36.

Reported on the linux-embedded ML



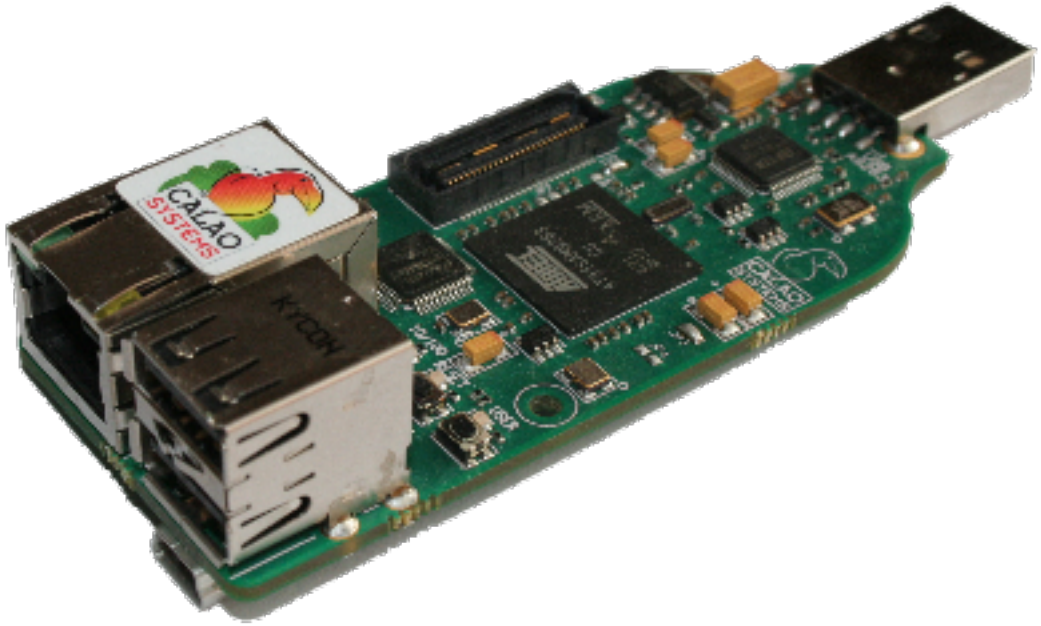
SquashFS

http://squashfs.sourceforge.net/

- Filesystem for block storage, so it doesn't support the MTD API.
- However, as it is read-only, it works fine with `mtdblock`, as long as the flash chip doesn't have any bad blocks.
- You could use it for read-only sections in your filesystem, but you cannot rely on it (bad blocks can always happen).

Benchmark hardware (1)

Calao Systems USB-A9263

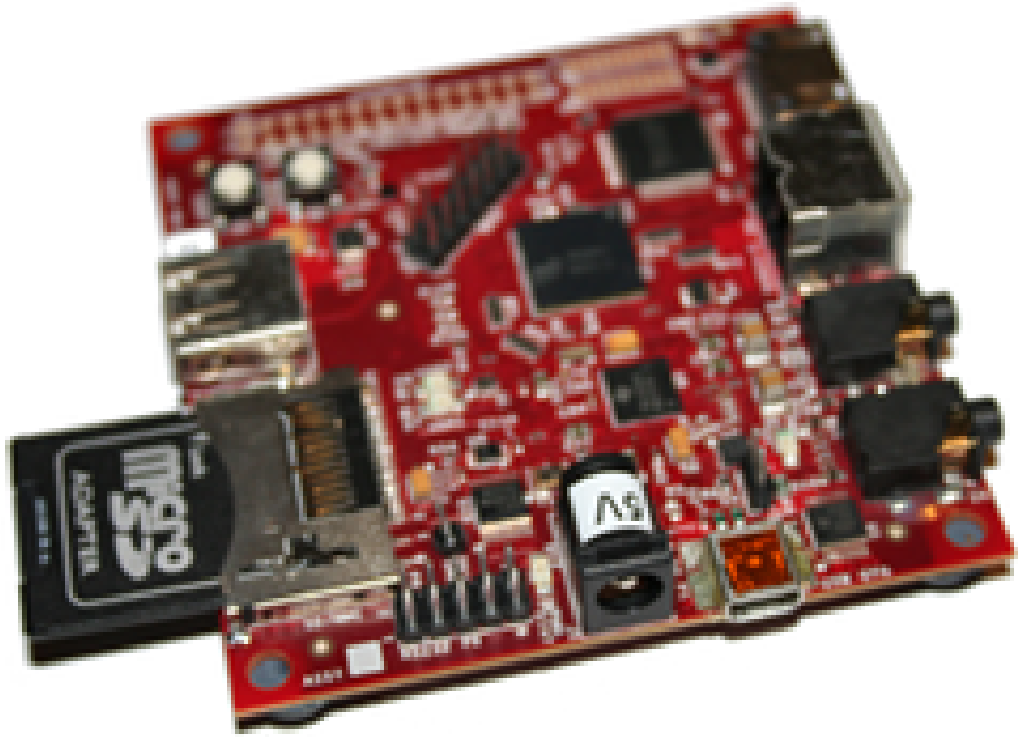


- AT91SAM9263 ARM CPU
- 64 MB RAM
  256 MB flash
- 2 USB 2.0 host
  1 USB device
- 100 Mbit Ethernet port
- Powered by USB!
  Serial and JTAG through this USB port.
- Multiple extension boards.
- Approximately 160 EUR

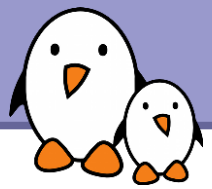
Supported in mainstream Linux since version 2.6.27!

Benchmark hardware (2)

TI Beagle Board



- TI OMAP 3530 ARM CPU
- 256 MB RAM, 256 MB flash
- RS-232 serial
- USB Host
  USB OTG
- JTAG
- DVI-D, S-Video
- Audio In and Out
- MMC/SD
- Only 150 USD



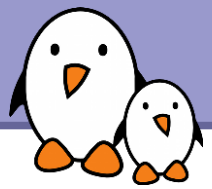
Benchmark methodology

Using a Debian Squeeze root filesystem for armel.

- Supports armv4 (Ubuntu on arm only supports v7)
- Contains all the tools to manipulate flash, as well as utilities for each filesystem.

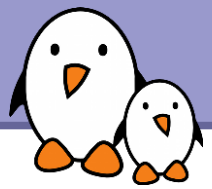
Using automated scripts supporting multiple boards

- Take care of sending commands to the boards through a serial line.
- Test rootfs and GPL scripts available on http://free-electrons.com/pub/utils/board-automation/

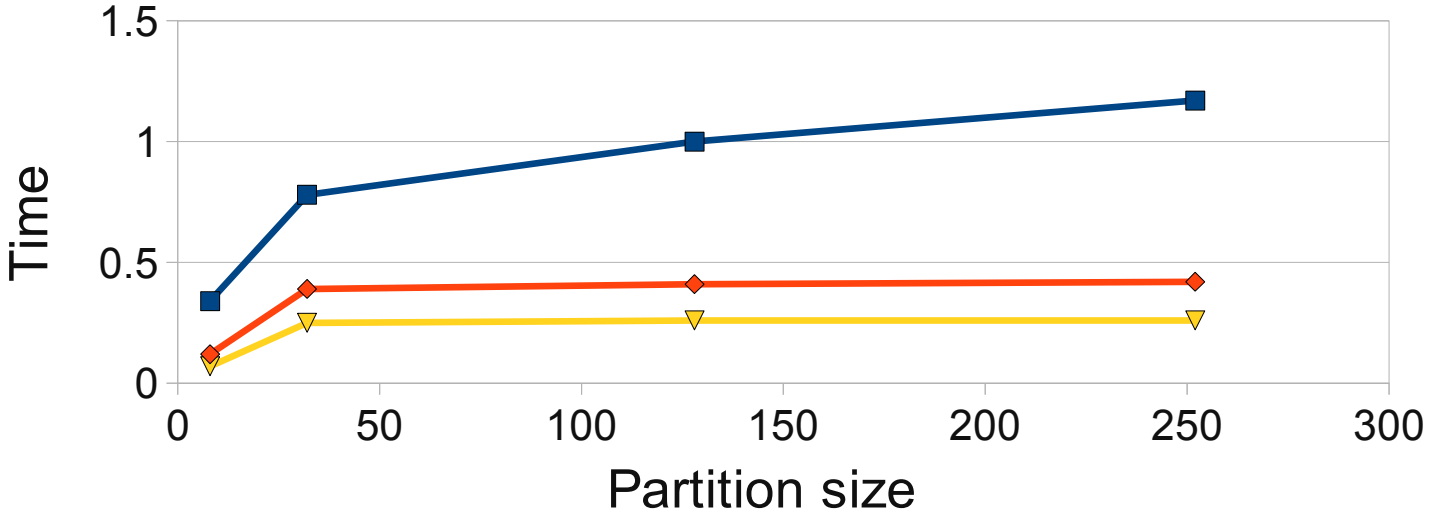


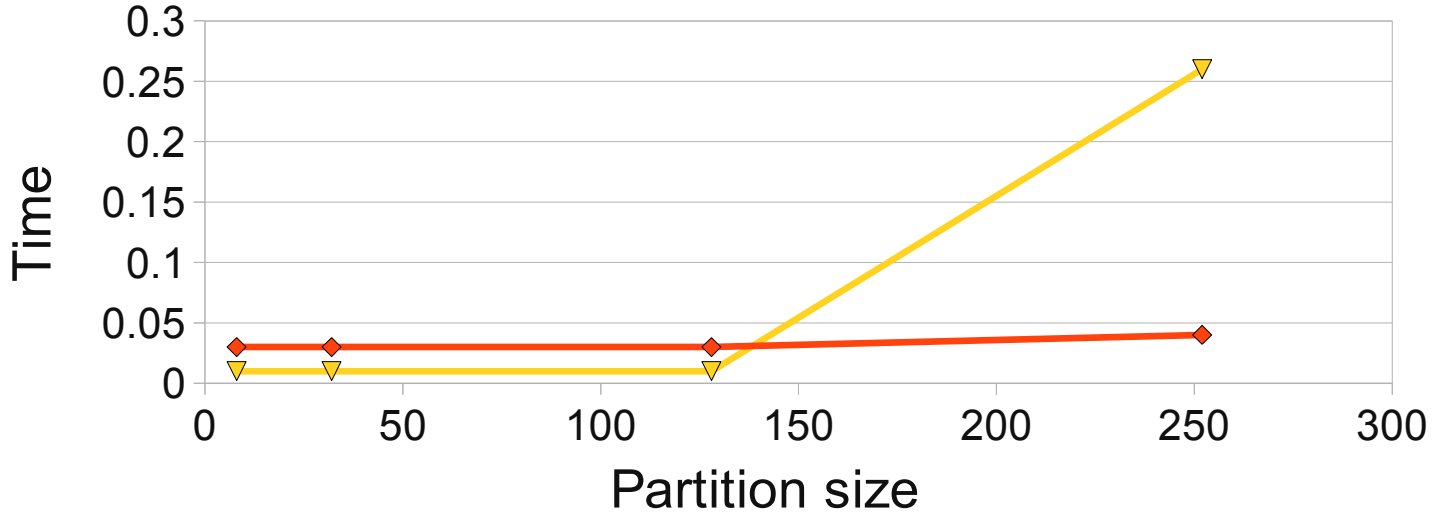
Initialization tests

- Time to load the filesystem drivers
- Special case of UBIFS:
  - ubi module initialization
  - ubiattach time
  - ubifs module loading time

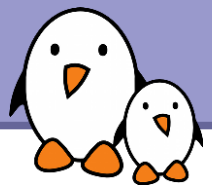


Initialization benchmarks





Note: couldn't measure ubifs on Beagle (bug)

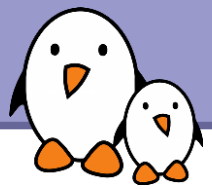


ubifs issue on Beagle

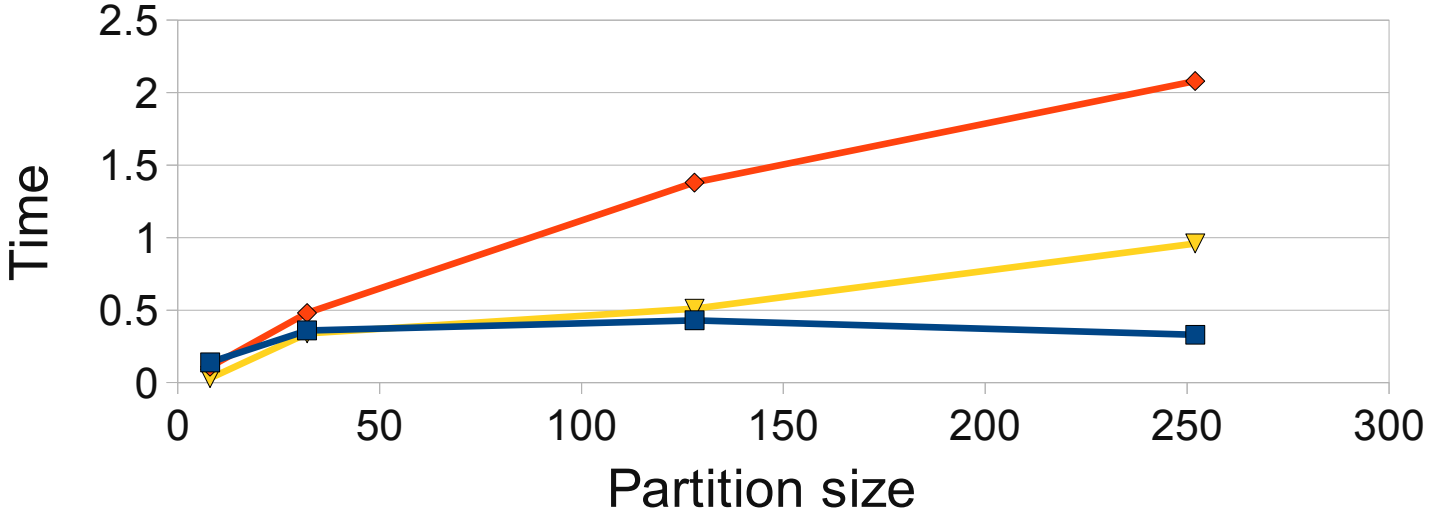
Methodology:

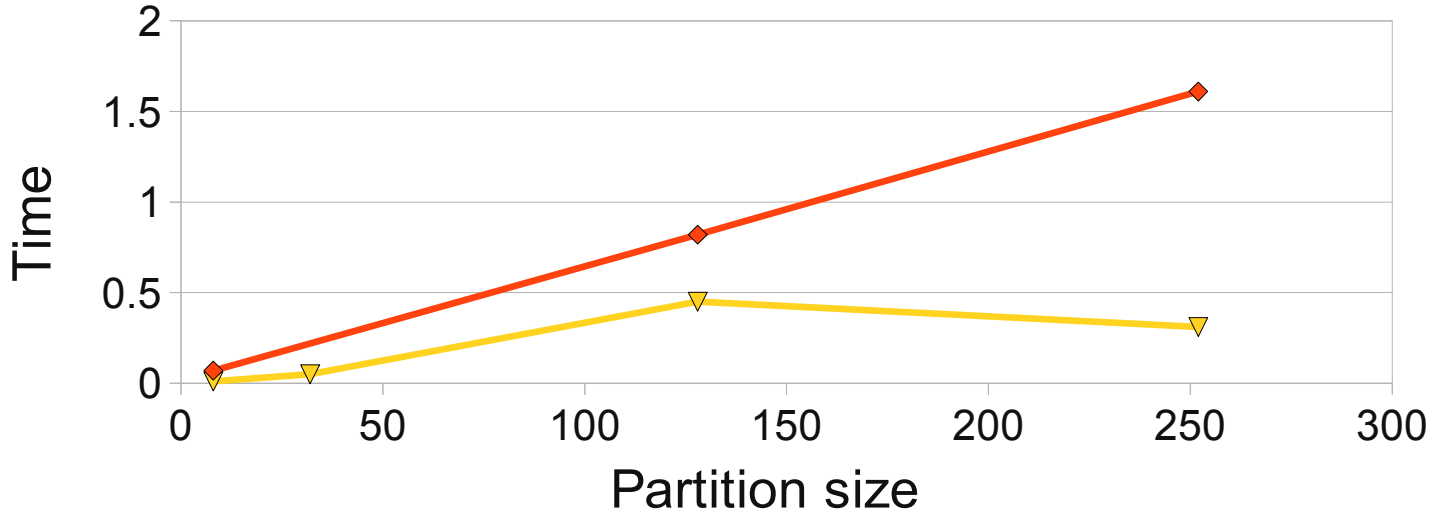
- ubiattach, mount, file, detach, attach again... oops
- No problem on Calao!

```
...
UBI error: ubi_io_read: error -74 (ECC error) while reading 64 bytes from
PEB 53:0, read 64 bytes
UBI error: ubi_io_read: error -74 (ECC error) while reading 64 bytes from
PEB 54:0, read 64 bytes
UBI error: ubi_io_read: error -74 (ECC error) while reading 64 bytes from
PEB 62:0, read 64 bytes
UBI error: ubi_io_read: error -74 (ECC error) while reading 64 bytes from
PEB 63:0, read 64 bytes
UBI warning: check_what_we_have: 29 PEBs are corrupted
corrupted PEBs are: 3 4 8 10 11 13 14 15 18 20 21 23 24 25 29 33 34 35 36 38
39 40 41 44 50 51 53 62 63
UBI error: check_what_we_have: too many corrupted PEBs, refusing this device
ubiattach: error!: cannot attach mtd1
```



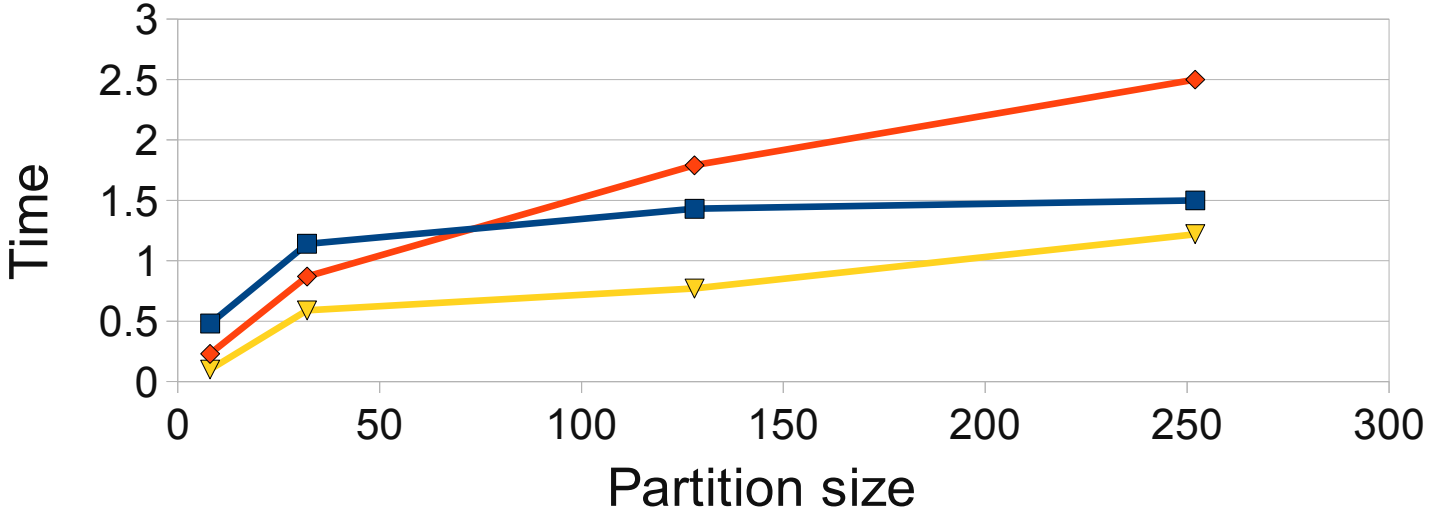
mount time benchmarks

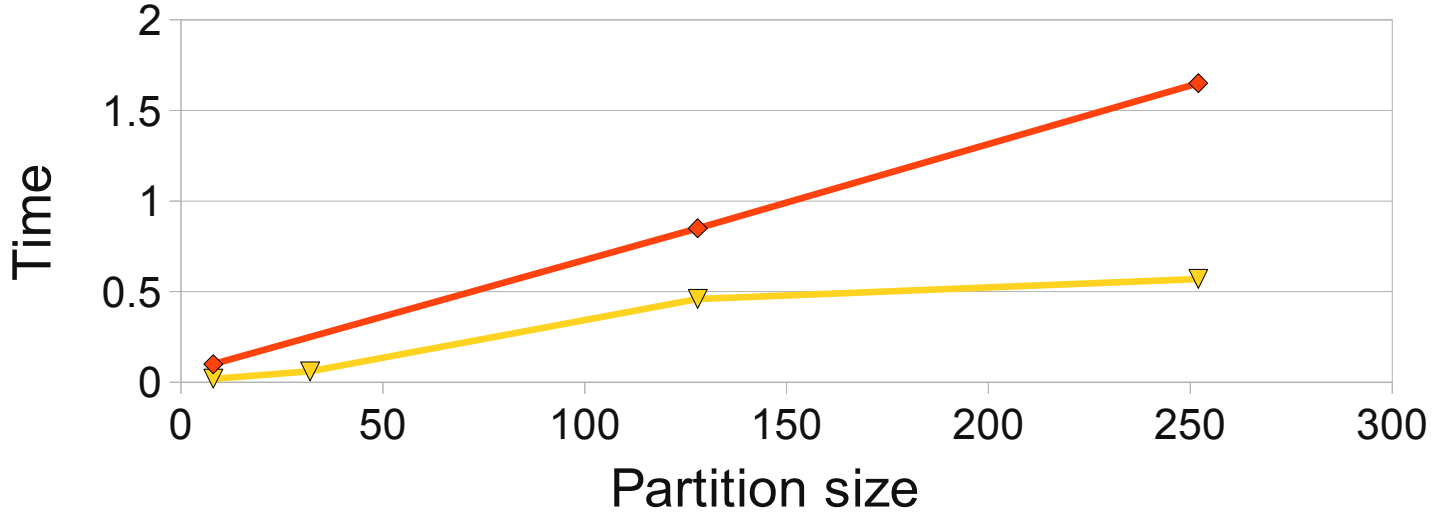




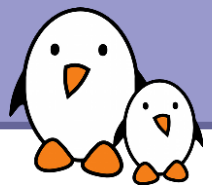
Note: couldn't measure ubifs on Beagle (bug)

init + mount time benchmarks

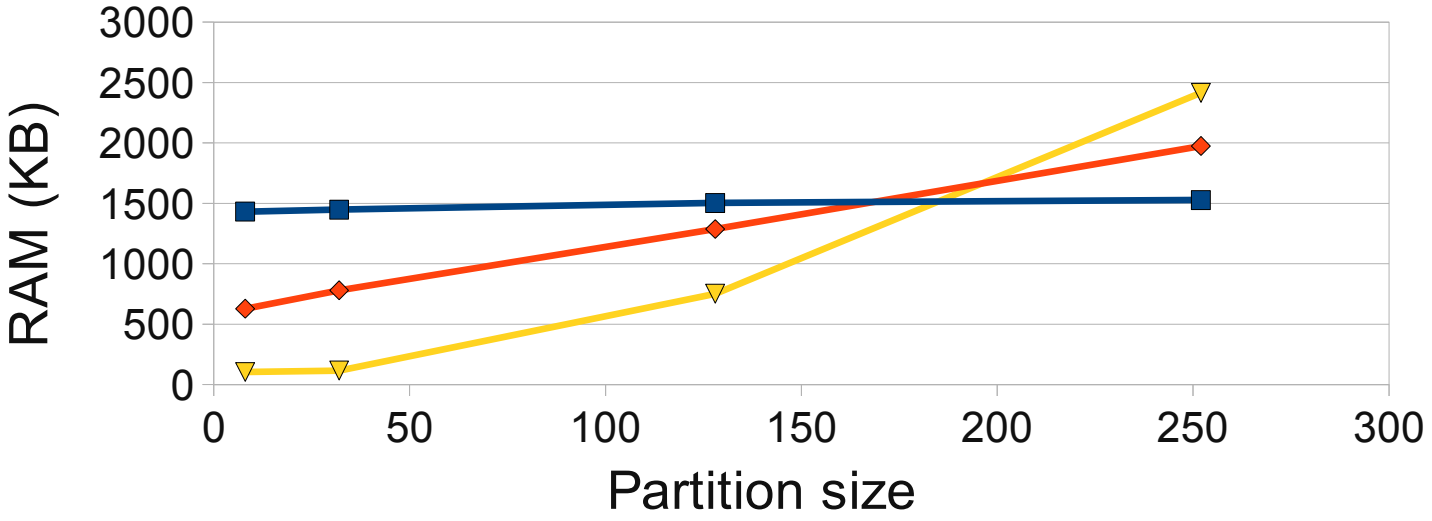


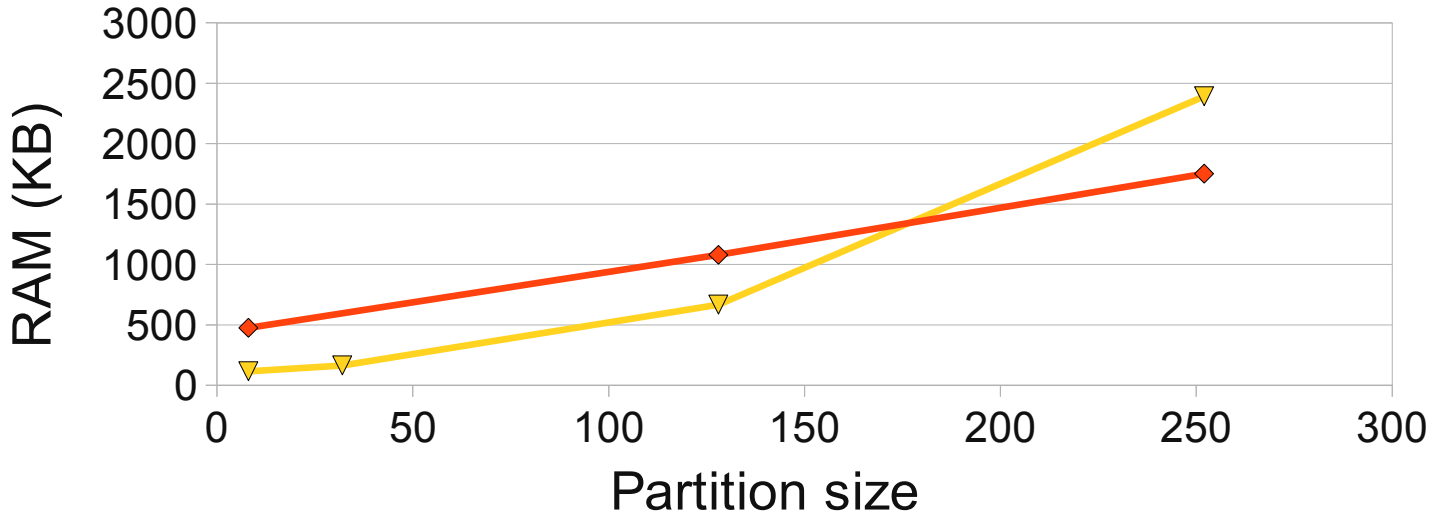


Note: couldn't measure ubifs on Beagle (bug)

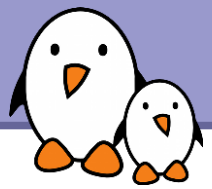


init + mount memory usage

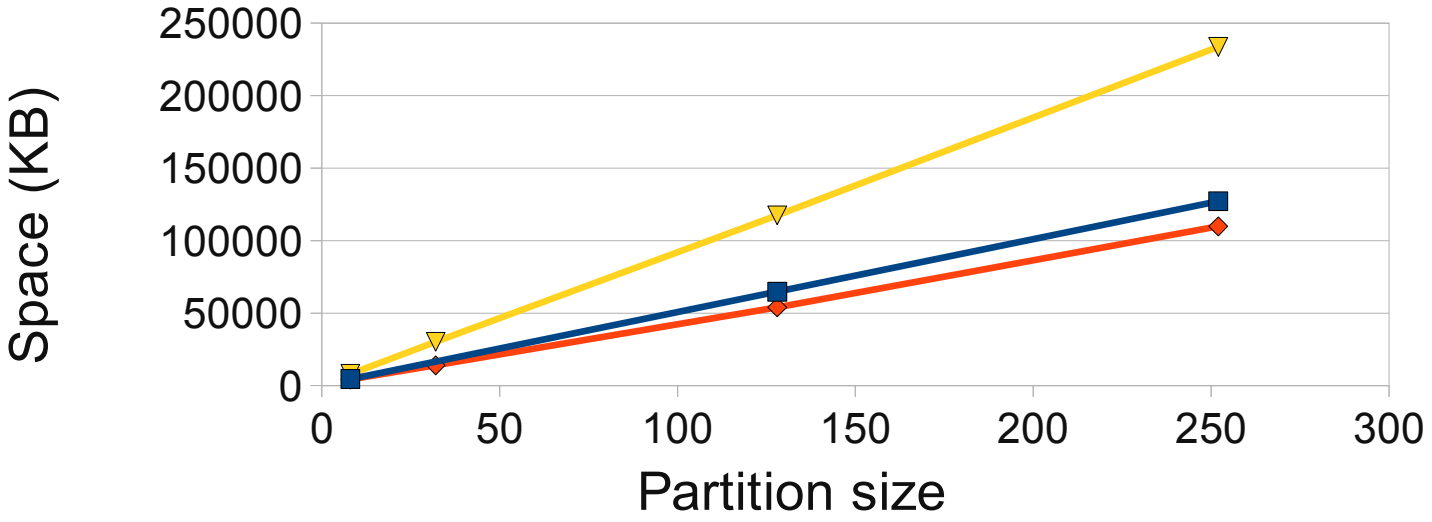


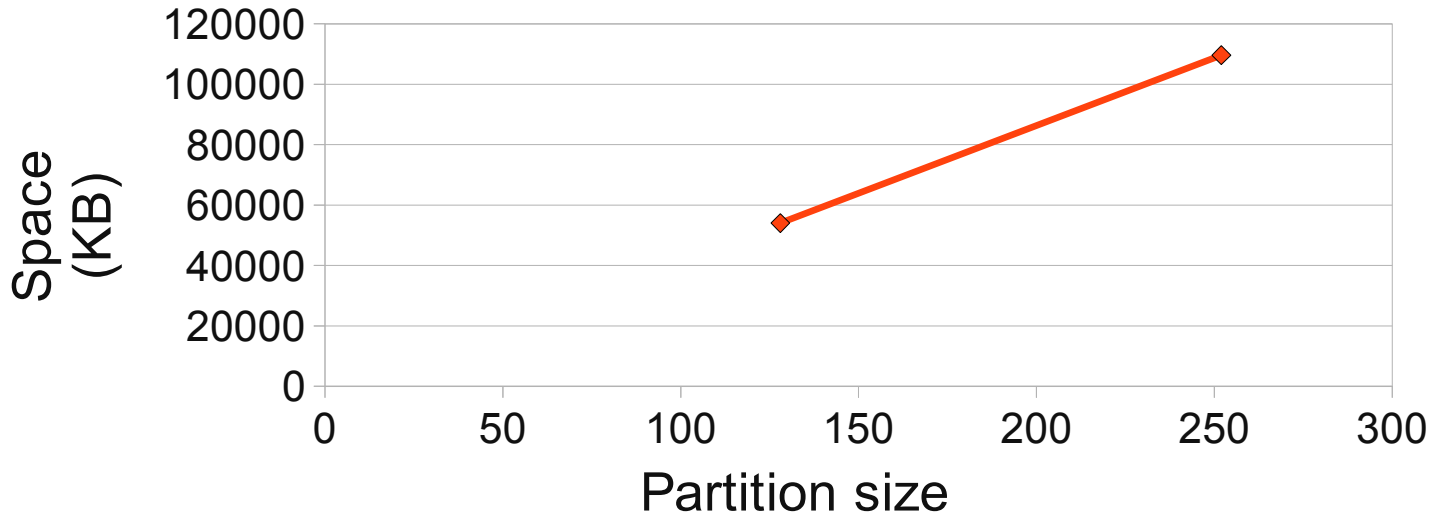


Note: couldn't measure ubifs on Beagle (bug)

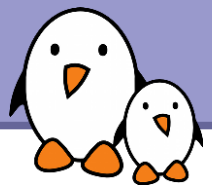


Used space

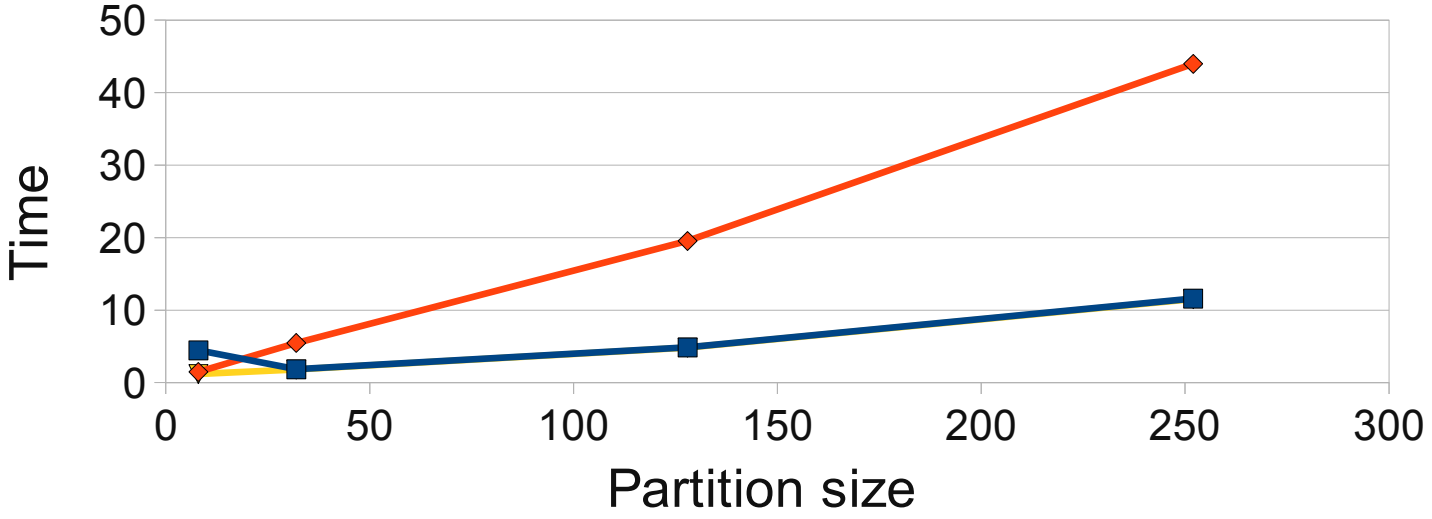


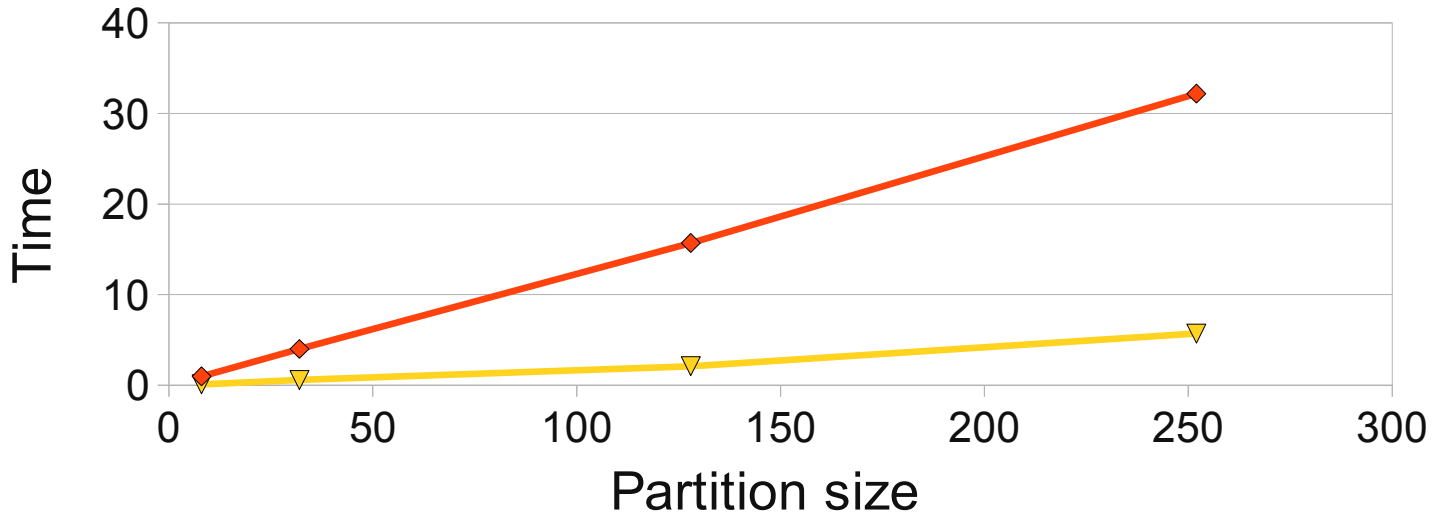


Note: data missing on Beagle (bug to investigate)

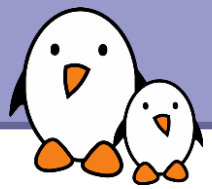


Read time benchmarks

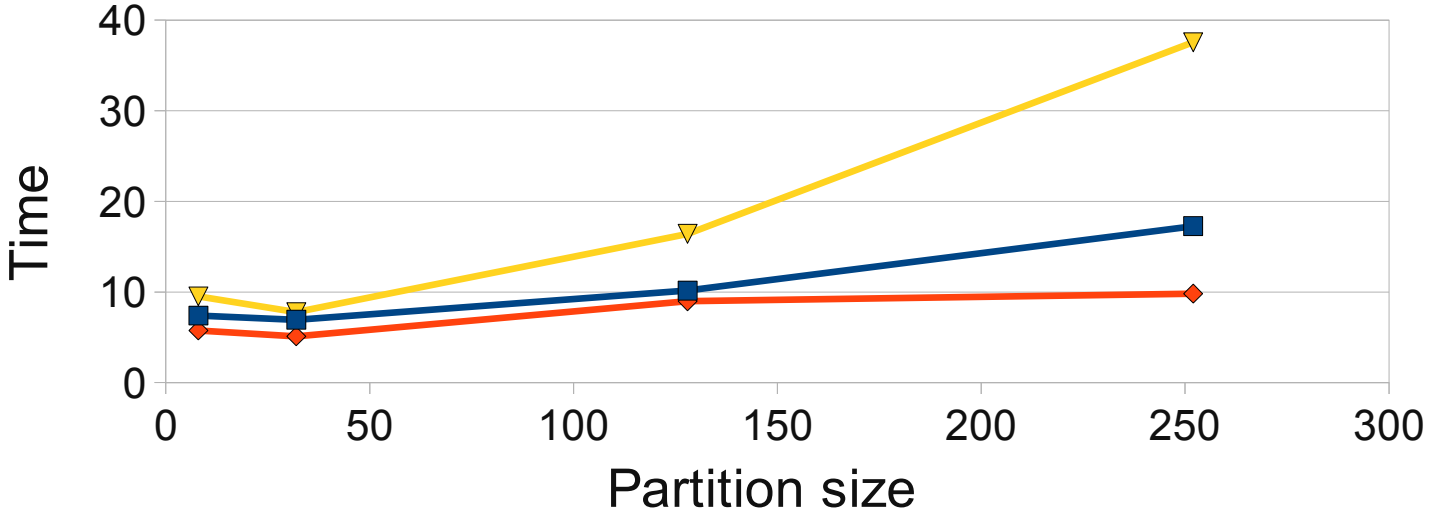


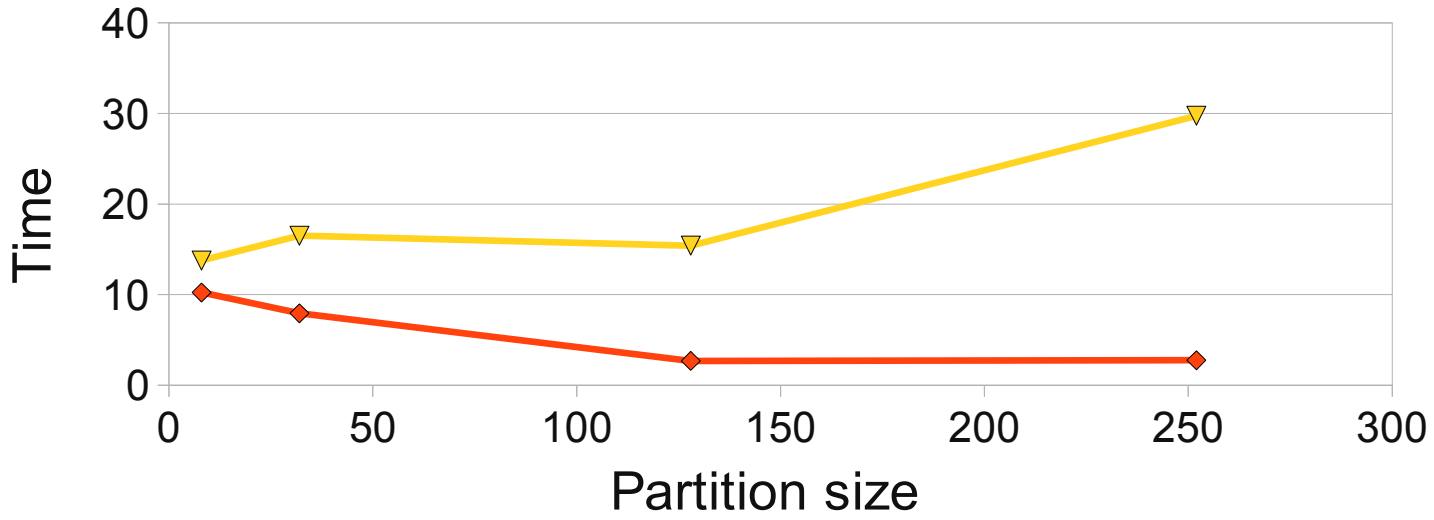


Note: couldn't measure ubifs on Beagle (bug)

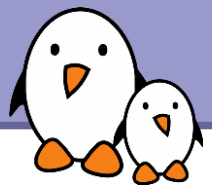


Remove time benchmarks

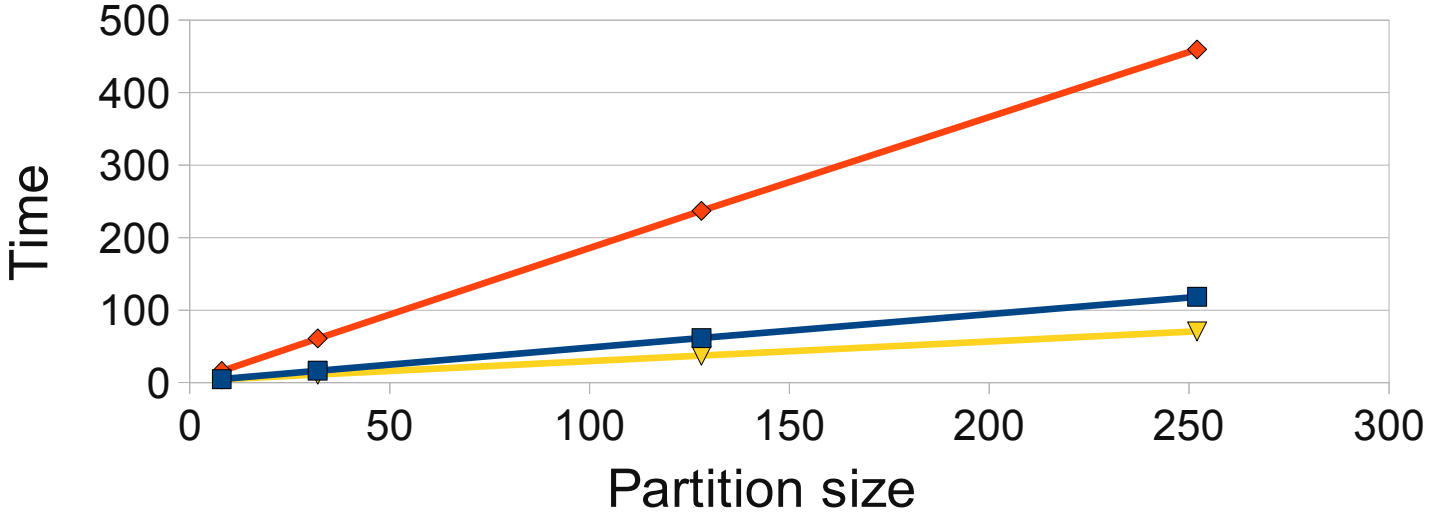




Note: couldn't measure ubifs on Beagle (bug)

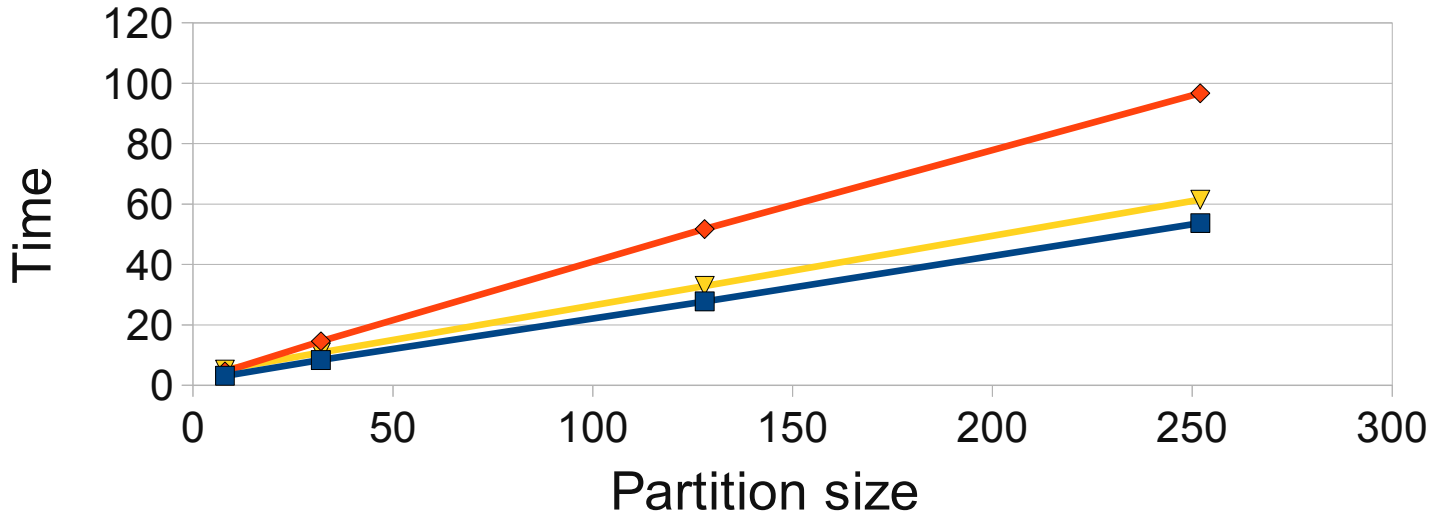


Write time benchmarks



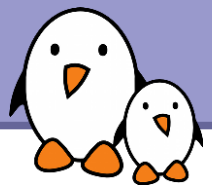
CALAO Board

ubifs
jffs2
yaffs2

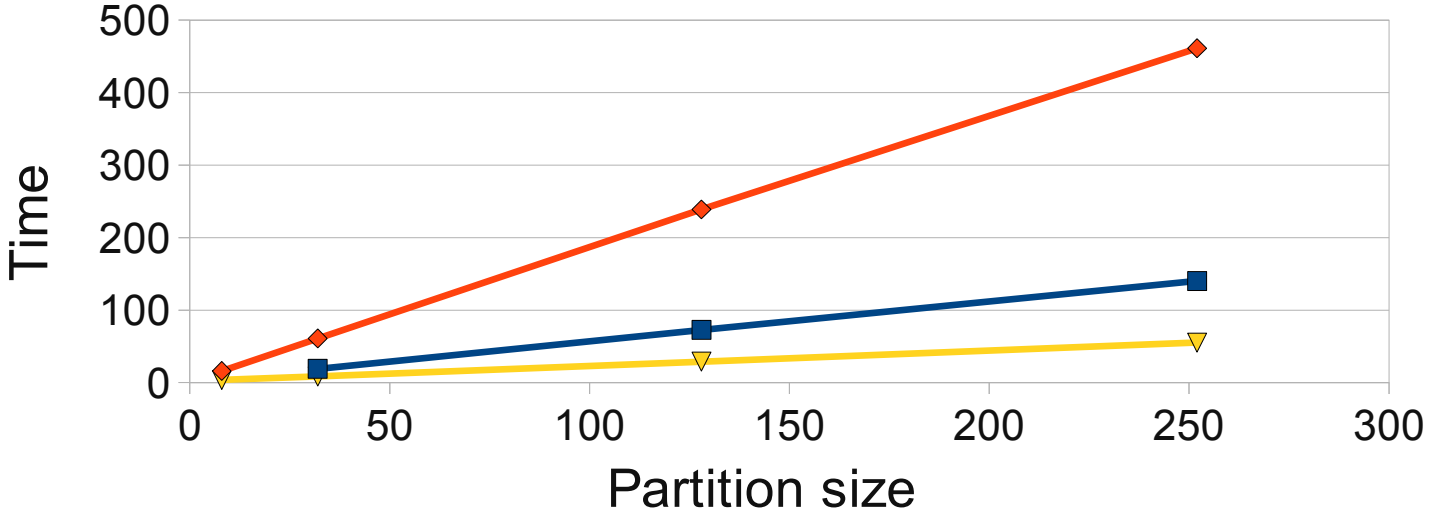


Beagle Board

ubifs
jffs2
yaffs2

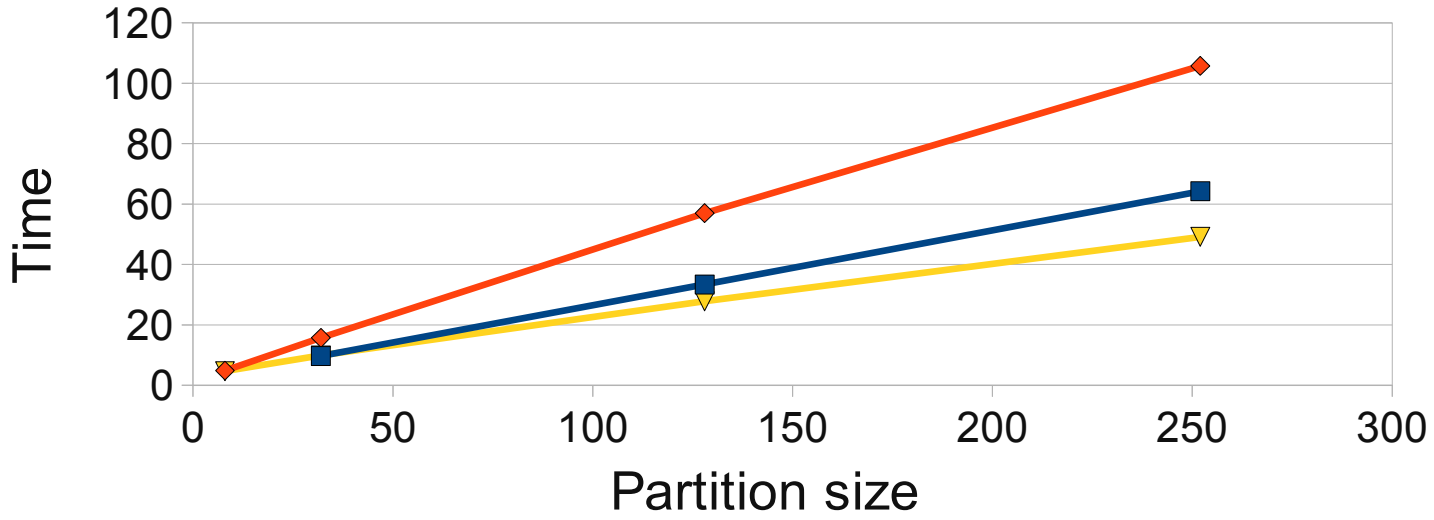


Video write time benchmarks



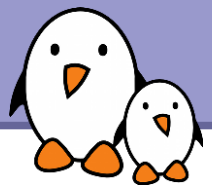
CALAO Board

ubifs
jffs2
yaffs2



Beagle Board

ubifs
jffs2
yaffs2



Flash filesystem conclusions

logfs

- Unusable so far. Crashes on both Beagle and Calao boards (2.6.36).

jffs2

- Poor performance compared to the others.
- May only make sense on small partitions where space matters.

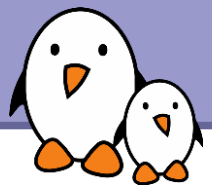
yaffs2

- Best choice for write performance.
- Good choice for small partitions when read/write performance and boot time matter more than space.
- Show get in mainline in the next months

ubifs

- Good or best performance in medium and big partitions

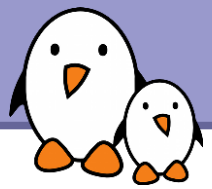
Time to replace your jffs2 partitions by ubifs or by yaffs2!



Future of MTD devices?

Embedded MMC (eMMC) starting to replace NAND flash

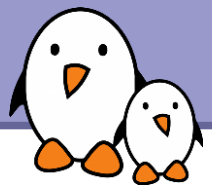
- Cheaper: ~30 EUR for 8 GB (Source: Calao Systems)
- Bad blocks managed internally
  No more issue with the bootloader sector going corrupt.
- Wear leveling could apply to the whole storage space
  (In theory, like with UBI. Not true with some devices).
- Automatic sleep mode
- Faster boot time: driver-less initialization.
- Can take advantage of block filesystems
  with optimizations for SSDs.
- How reliable, trustworthy and resistant are these?
  Loosing control on wear leveling.



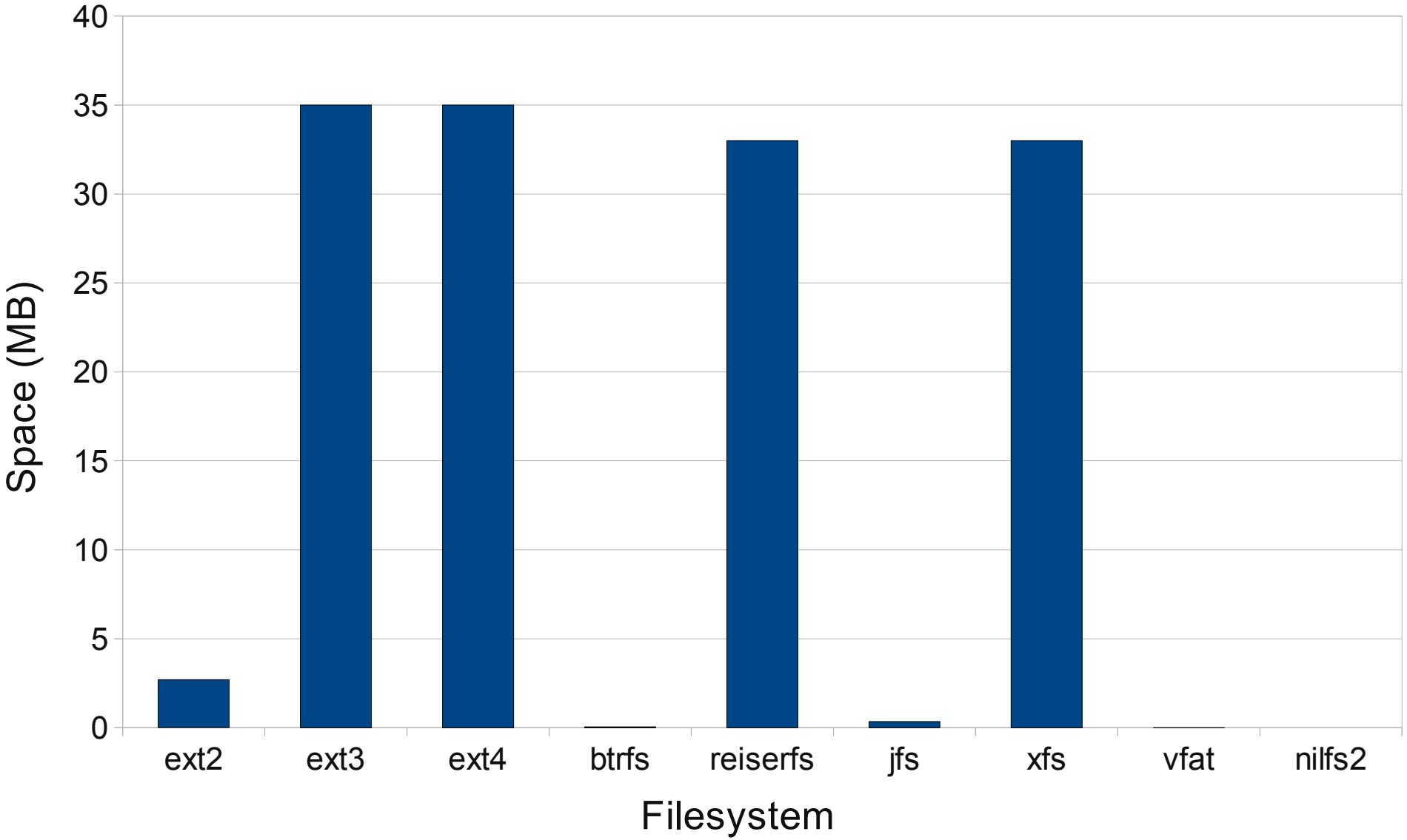
Quick block fs benchmarks

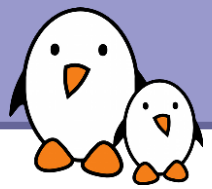
Methodology

- Tests run on the Beagle board
- Copy 800 MB of rootfs files from USB to MMC
- Flush caches
- Read MMC contents
- Flush caches
- Remove MMC contents
- Write 100 copies
  of a 6 MB video to MMC

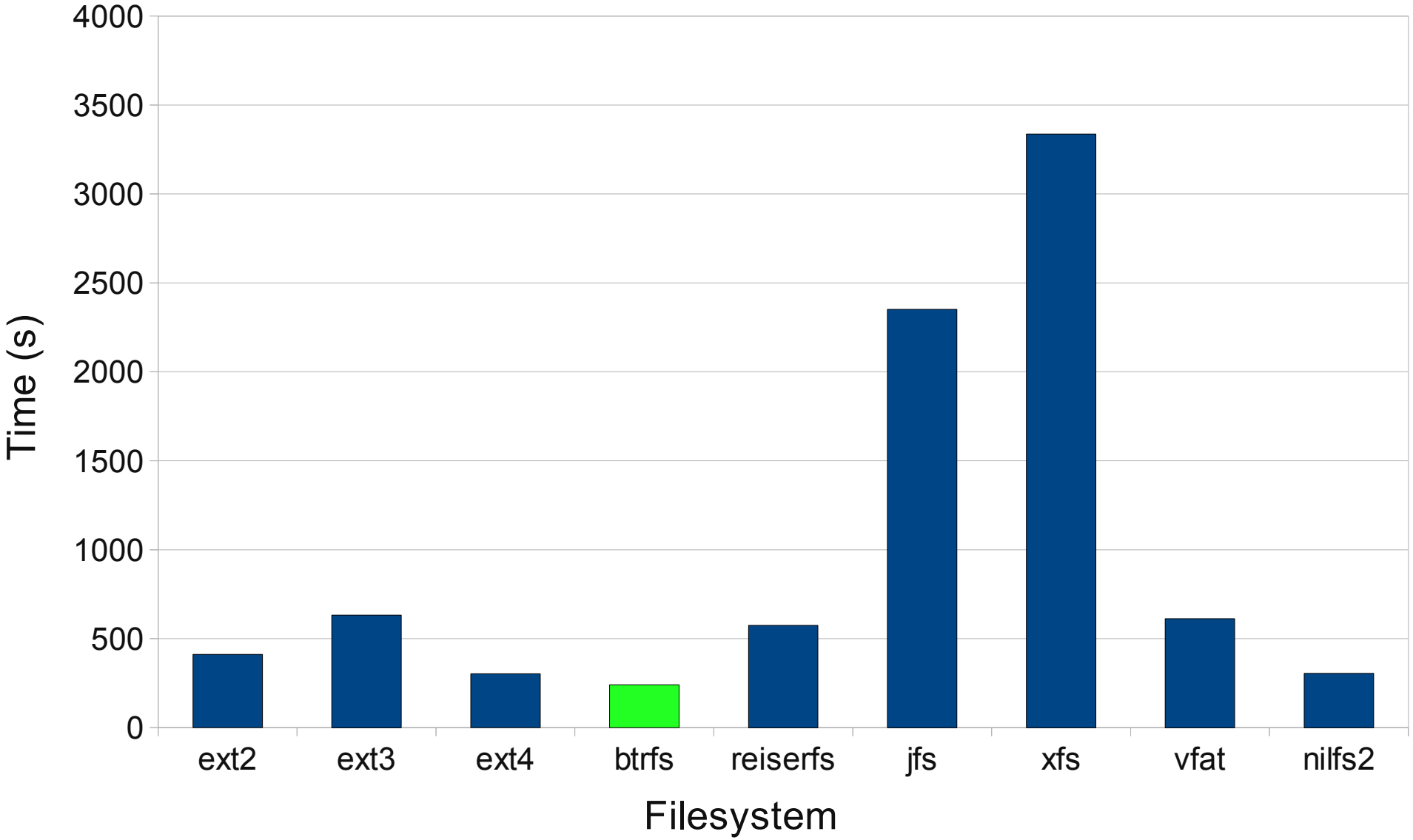


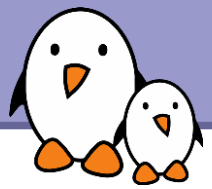
Disk usage after format



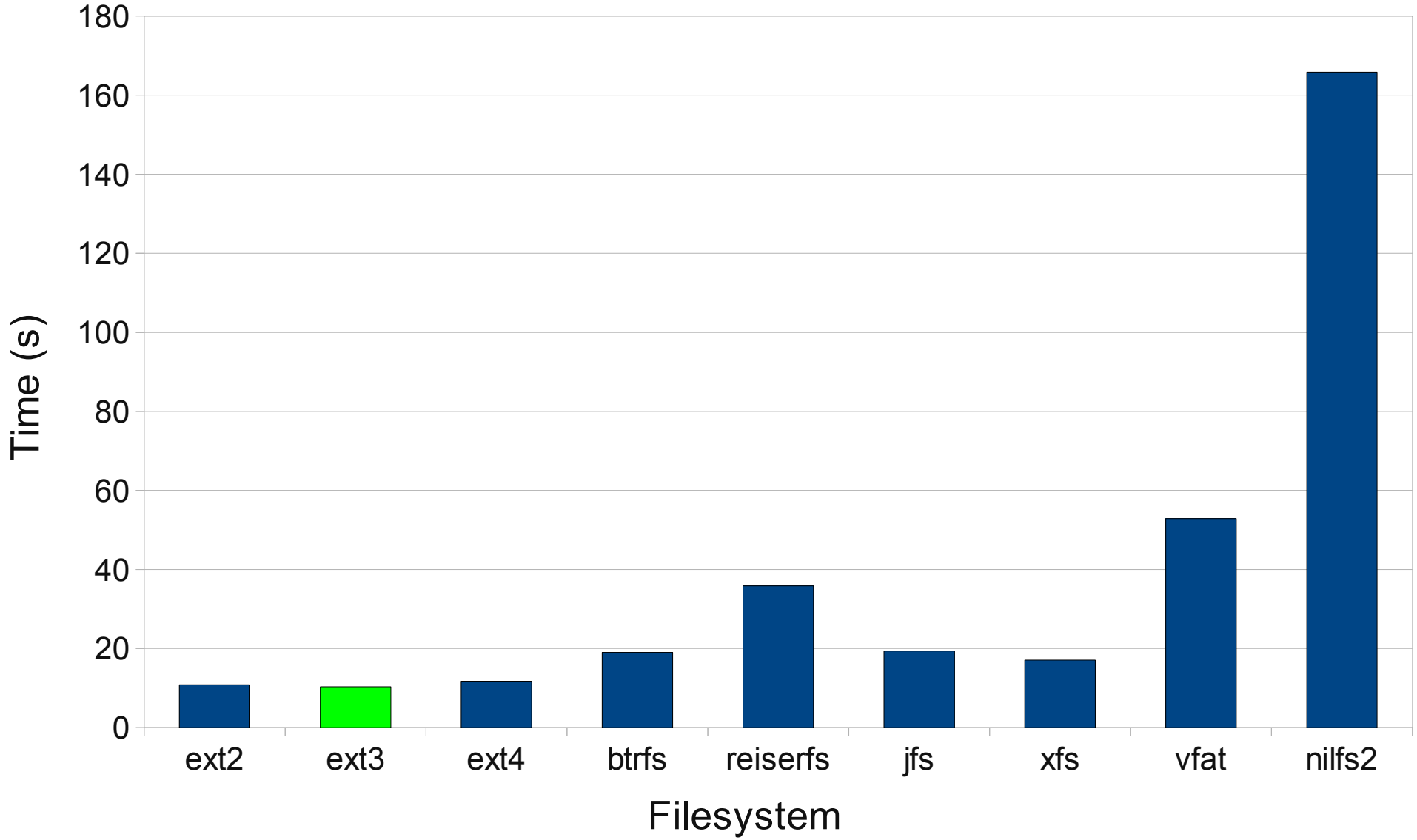


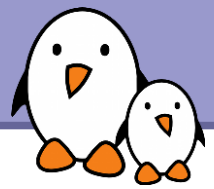
MMC copy from USB (ext3)



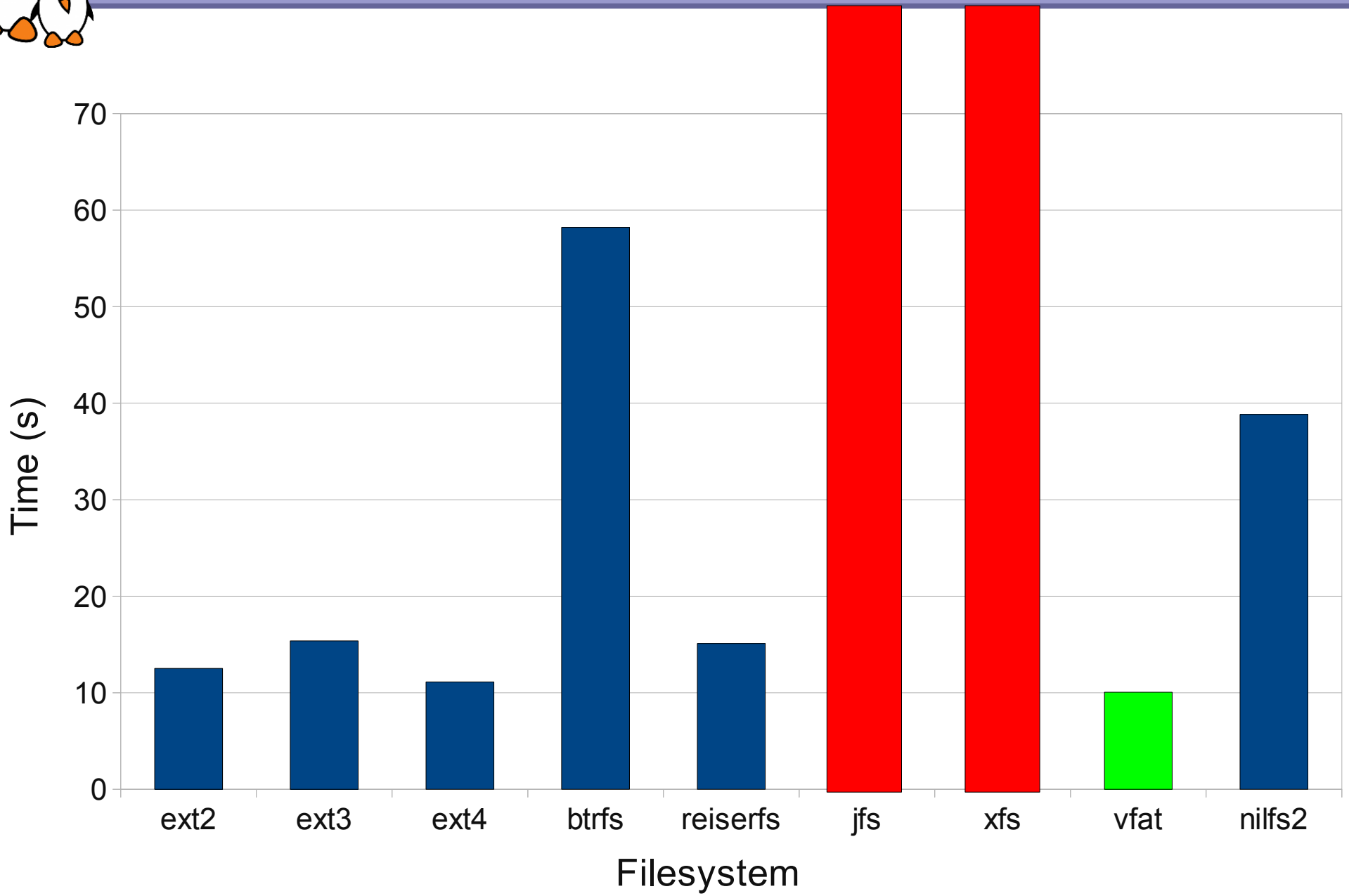


MMC read time

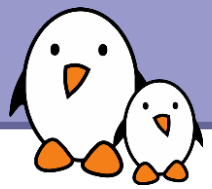




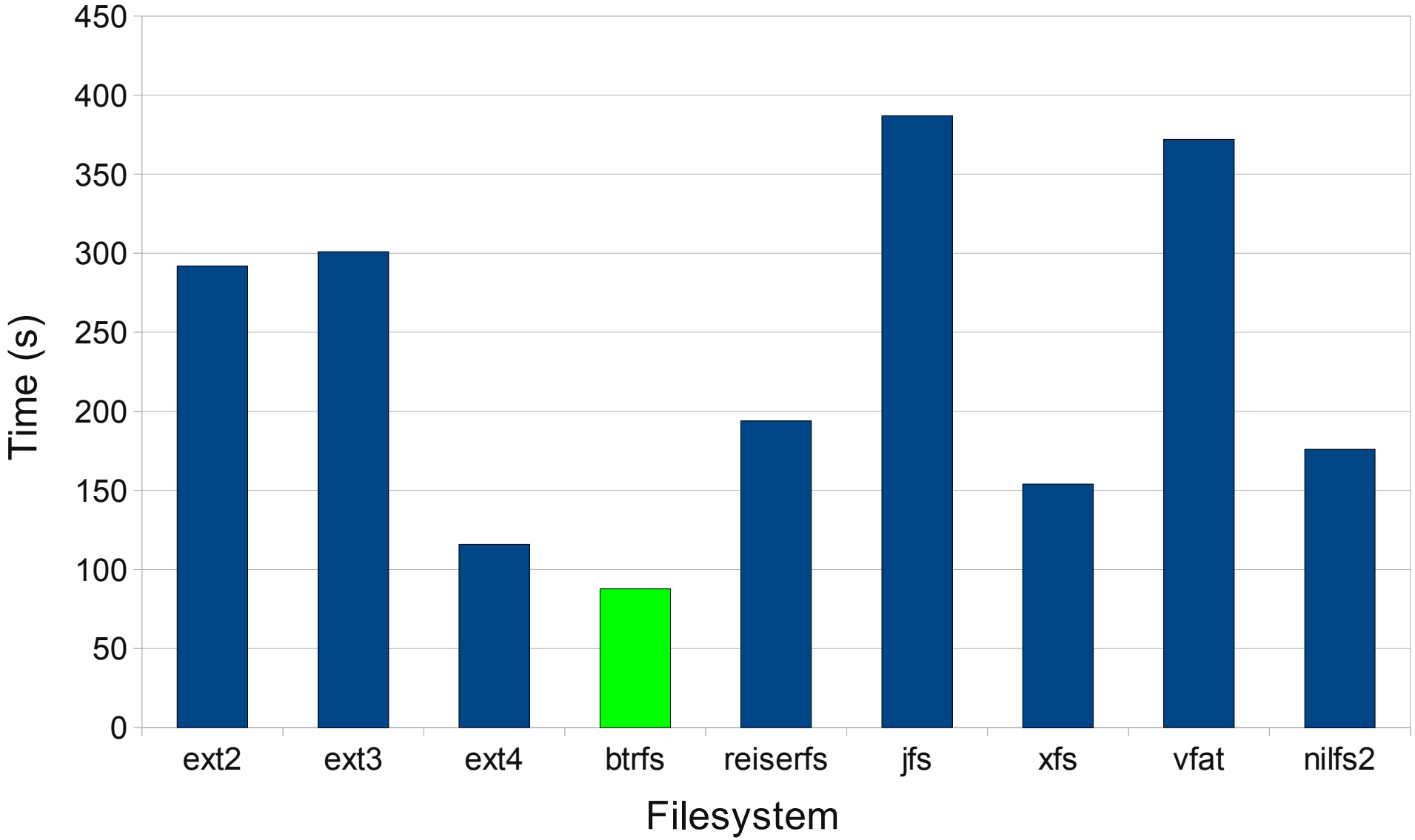
MMC remove time

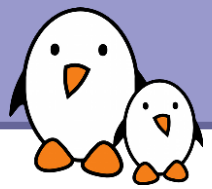


jfs: 2872s! xfs: 239s!

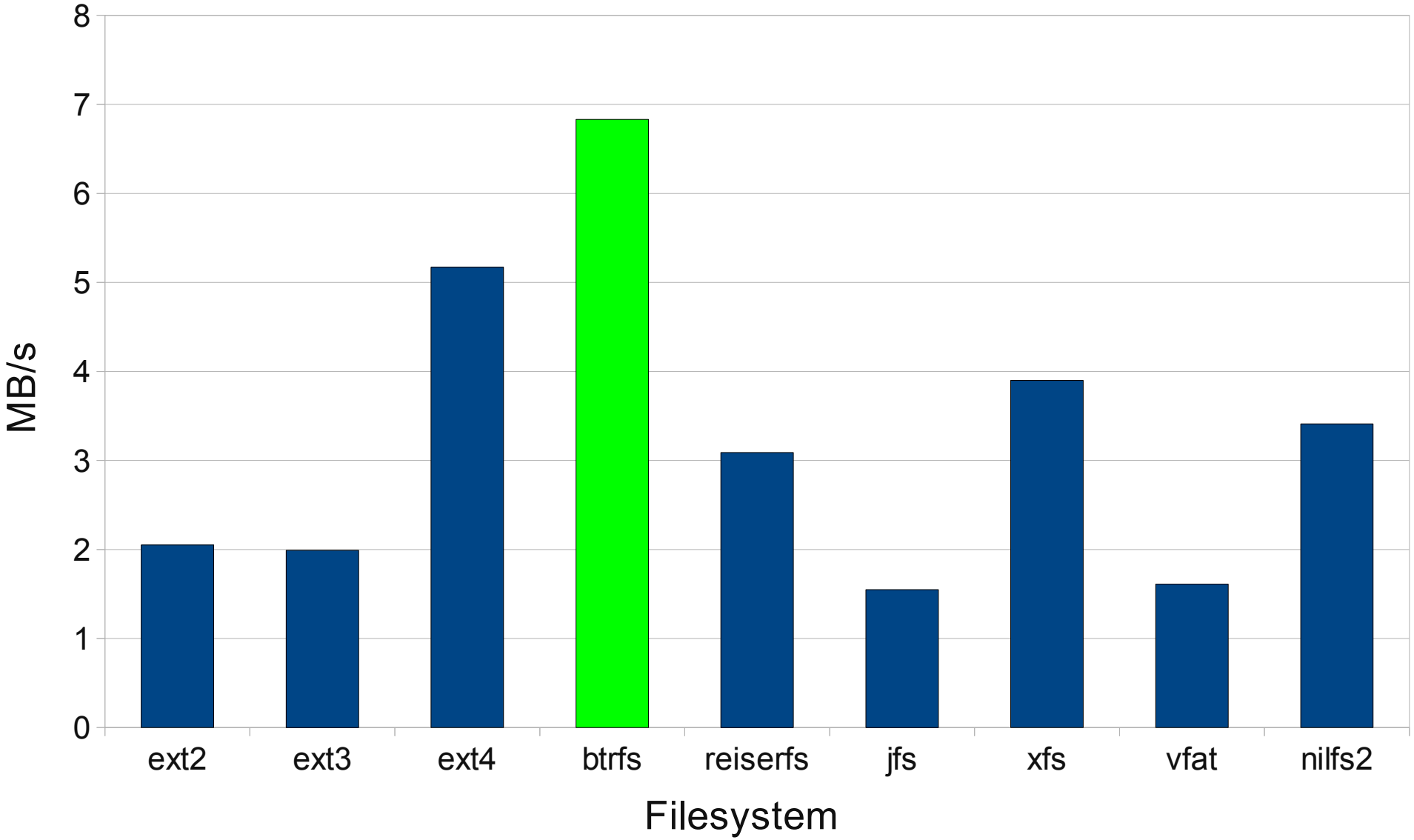


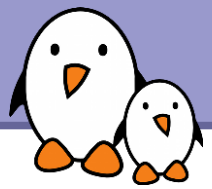
MMC write video time





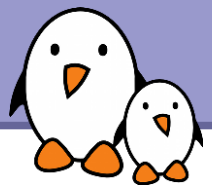
MMC video write speed (MB/s)





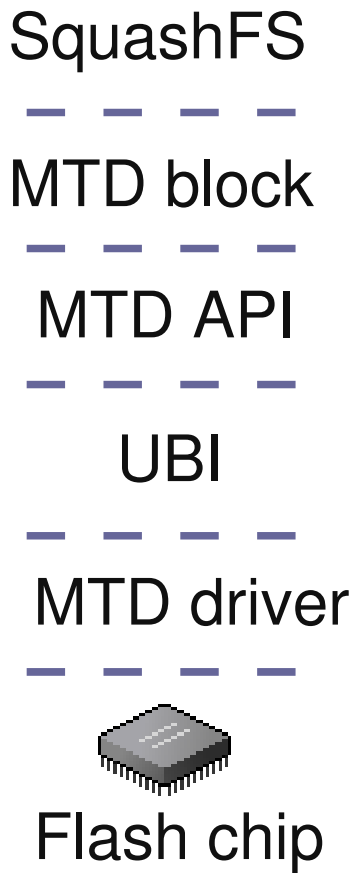
block storage: conclusions

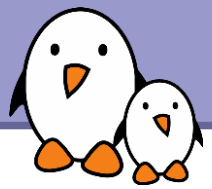
- ext4 is great!
  Best compromise between maturity and performance.
- btrfs rocks!
  Though not production ready (still experimental).
- xfs is very disappointing
  (it has good performance on rotating disks)
- Hard to tell which filesystem will work best on your system.
  Make your own experiments (switching filesystems is cheap)!



Call for change

- Squashfs on top of UBI
  Too much overhead today
- Block device on top of UBI
- Yaffs2 in mainline!
- Merge the Logfs Forum with the MTD Foundation ;-)



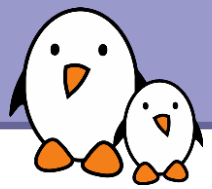


Thanks!

Questions?

Comments?

Scripts and test root filesystem on
http://free-electrons.com/pub/utils/board-automation/



Flash fs mini-BOF

- Do you think that eMMC will completely replace NAND flash?
- Would you feel comfortable to use block flash storage as a swap area?
- How do you limit the number of writes in a read/write partition?
- Any project you would like CELF to support?