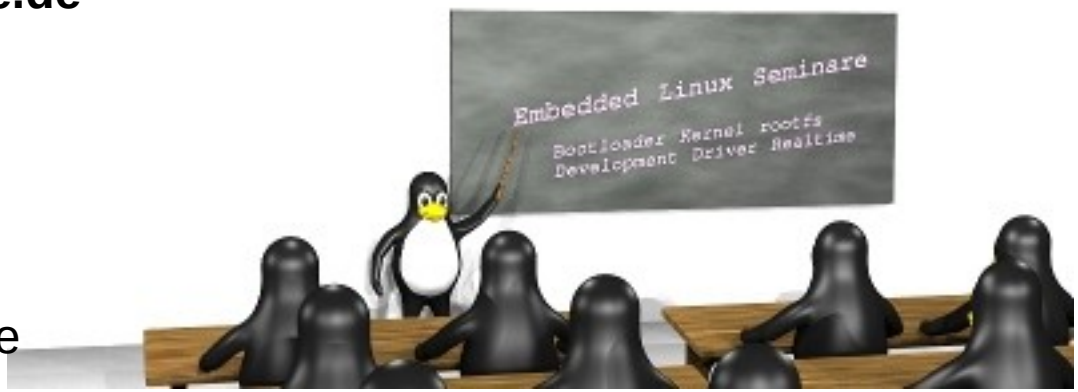
Embedded-Linux-Seminare

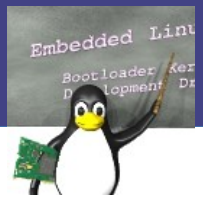
File Systeme

http://www.embedded-linux-seminare.de

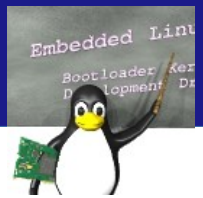
Diplom-Physiker Peter Börner
Spandauer Weg 4
37085 Göttingen

Tel.: 0551-7703465
Mail: info@embedded-linux-seminare.de



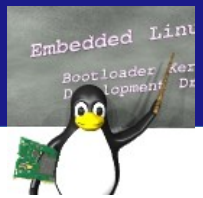


Kopier-Rechte



Inhalt

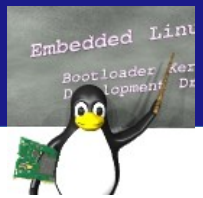
- Einführung
- Unterstützte Filesysteme
- ext2/ext3/ext4
- jffs2
- ubifs



Einführung

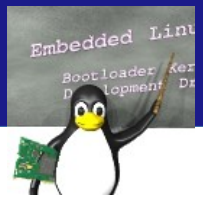
- Filesysteme sollen
  - Daten sicher speichern und verwalten
  - Auf verschiedenen Medien einsetzbar sein
  - Das virtuelle Filesystem unterstützen

- Unterstütze Speichermedien
  - Festplatten
  - Cds/DVDs
  - Flash
  - Netzwerk
  - Arbeitsspeicher
  - ...



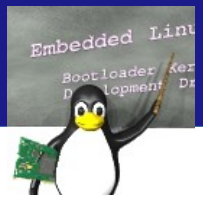
Einführung

- Unterstützte Features
  - Verschlüsselung
  - Journaling
  - Nur Lesbarer Zugriff
  - Komprimierung



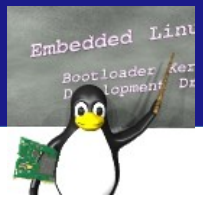
Einführung

- Journaling
  - Filesystem Treiber bündeln Schreibzugriffe auf Medium, um Performance zu verbessern.
  - Bei Power-Failure vor realem Schreibzugriff auf Medium gehen Daten verloren
  - Journaling Filesysteme tragen Schreibzugriffe in ein Journal ein
    - Nur Metadaten werden gesichert
    - Metadaten und echte Daten werden gesichert
  - Sollen konistentes Filesystem nach einem Power-Failure sicherstellen.
  - Beim Neustart werden alle Transaktionen, die im Journal verblieben sind ausgeführt.
  - Benötigt Overhead
  - Verlangsamt je nach Konfiguration den Schreibzugriff mehr oder minder stark.



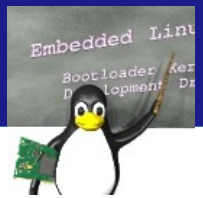
Unterstützte Filesystem

- Linux unterstützt zur Zeit über 50 Filesysteme
- Filesysteme für „normale“ Festplatten:
  - ext2/ext3/ext4
  - FAT/VFAT
  - NTFS
  - HFS
  - Minix
  - …
- Filesysteme für CD/DVD
  - ISO9660
  - UDF (Universal Disk Format)



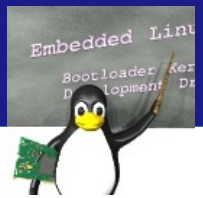
Unterstützte Filesysteme

- Filesysteme für direkten Zugriff auf Flash Speicher
  - JFFS/JFFS2 (Journaling Flash Filesystem)
  - YAFFS (Yet Another Flash Filesystem)
  - UBIFS (Unsorted Block Images)
- Filesysteme für RAM
  - Cramfs (Compressed RAM Filesystem)
  - SquashFS
  - Tmpfs
  - Initramfs



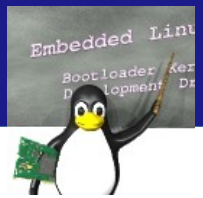
Unterstützte Filesysteme

- Filesysteme für Netzwerkzugriff
  - NFS (Network File System)
  - cifs (Windows Shares)
  - GCFS, OCFS (Cluster File Systeme)
- Pseudo File Systeme
  - procfs (Process File System)
  - sysfs (System File System)



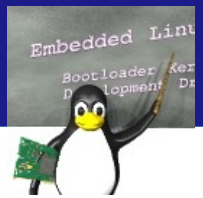
Einbinden von Filesystemen

- Mounten
  - Filesysteme müssen per mount Befehl an einer Stelle im virtuellen Filesystem eingehängt werden
    - `mount -t <type> -o <optionen> <device> <mountp>`
      z.B.
      ```
      mount -t ext4 /dev/sda1 /
      mount -t nfs server:/home/tux /home/tux/extern
      mount -t auto /dev/sdb1 /media/usb-stick
      mount -t iso9660 -o loop /tmp/cd.img /media/cd
      ```
  - Wenn der Zugriff nicht mehr benötigt wird kann das Filesystem wieder ausgehängt werden
    - `umount /home/tux/extern`
    - **Achtung:** Der Zugriff auf das Filesystem muß beendet, also z.B. alle Dateien wieder geschlossen sein.
    - Der umount Befehl sorgt dafür, dass alle im Schreib-Cache verbliebenen Daten auf das Medium geschrieben werden.
      Wichtig vor dem Entfernen von z.B. USB Medien



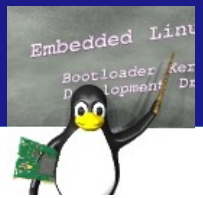
Flash Filesysteme

- Flash Filesysteme sollen den direkten Zugriff auf Flashspeicher ermöglichen.
- Wichtigstes Feature: Wear leveling zum optimalen Verteilen der Anzahl der Schreibzugriffe auf alle Zellen.
- Flash Bausteine sind in relativ große Erase-Blocks aufgeteilt. z.B. 128 kB bei NOR Flash Chips.
  - Löschen der Blocks setzt alle Bits auf 1.
  - Schreiben von Daten bedeutet setzen der Bits mit 0 Wert.
  - Vor dem Schreiben neuer Daten muss der entsprechende Block gelöscht werden.
  - Löschen der Blöcke belastet die Lebensdauer der Zellen (typischerweise 100000 Zyklen)
- USB Sticks, CF-Cards und SD-Cards haben einen Hardware-Controller. Sie können mit normalen Filesystemen verwendet werden.



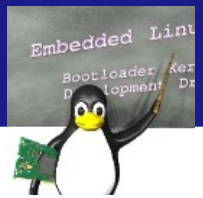
JFFS2

- Journaling Flash File System 2
- Ersetzt das veraltete JFFS.
- Beinhaltet Journaling und Wear-Leveling
- Beim Mounten wird der ganze JFFS2 Speicherbereich gescannt, um festzustellen welcher Block zu welcher Datei gehört.
  - Lange Bootzeiten
  - CONFIG_JFFS2_SUMMARY Kernel Option erlaubt das Sichern dieser Information.
  - Muss beim Formatieren mit angegeben werden.
  - Die Bootzeit verkürzt sich (z.B. von 16s auf 0,8s bei 128 MB)
- Unterstützt NOR und NAND Flash
- Unterstützt Komprimierung



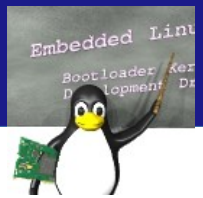
JFFS2

- Log strukturiertes Filesystem
  - Änderungen werden in Nodes gelogged
  - Mehrere Nodes pro Erase-Block
  - Nodes beinhalten die (komprimierten Metadaten)
- Besitzt einen Müll-Sammler (Garbage-Collector)
  - Wear-Leveling schiebt die Daten sukzessive über die Partition
  - Garbage-Collector fügt leere und nicht mehr benutzte Nodes zu leeren Blöcken zusammen.
  - Leere Blöcke werden dann gelöscht und stehen wieder zur Verfügung
  - Problem bei Power-Failure, da der Prozess im Hintergrund arbeitet.



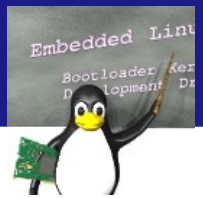
JFFS2

- Formatieren
  - mkfs.jffs2
  - Parameter erläutern



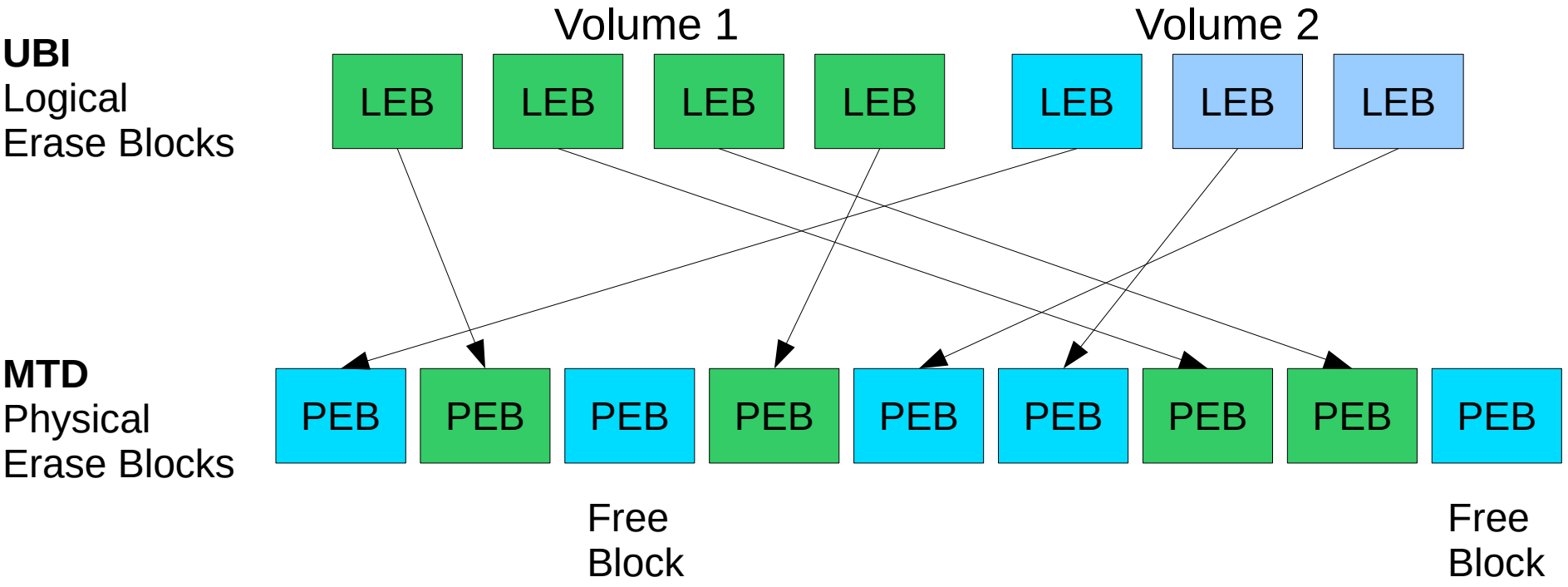
YAFFS

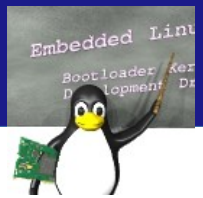
- Yet Another Flash File System
  - Speziell für NAND Flash designed
  - Als performanterer Ersatz für JFFS2 gedacht
  - Keine Komprimierung
  - Wear Leveling
  - Power Failure sicher
  - Schnellere Bootzeit
  - Log-Strukturiert
- Formatierung
  - mkfs.yaffs



UBIFS

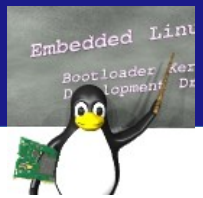
- UBI Unsortiertes Block Image
  - Volume Management System auf MTD Layer aufgesetzt
  - Erlaubt die Erzeugung mehrerer logischer Volumes.
  - Ermöglicht Schreibzugriffe über alle physikalischen Blöcke
  - Kümmert sich um Erase Blocks und Wear leveling





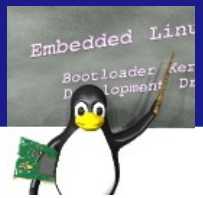
UBIFS

- UBIFS
  - Setzt auf UBI auf
  - Ab Kernel 2.6.27 implementiert
  - Overhead für Metadaten nachteilig für kleine Partitionen
- Als Ersatz für JFFS2 gedacht
- Kürzere Mount-Zeiten
- Robuster gegenüber Power-Failure
- CRC und ECC für Daten Integrität



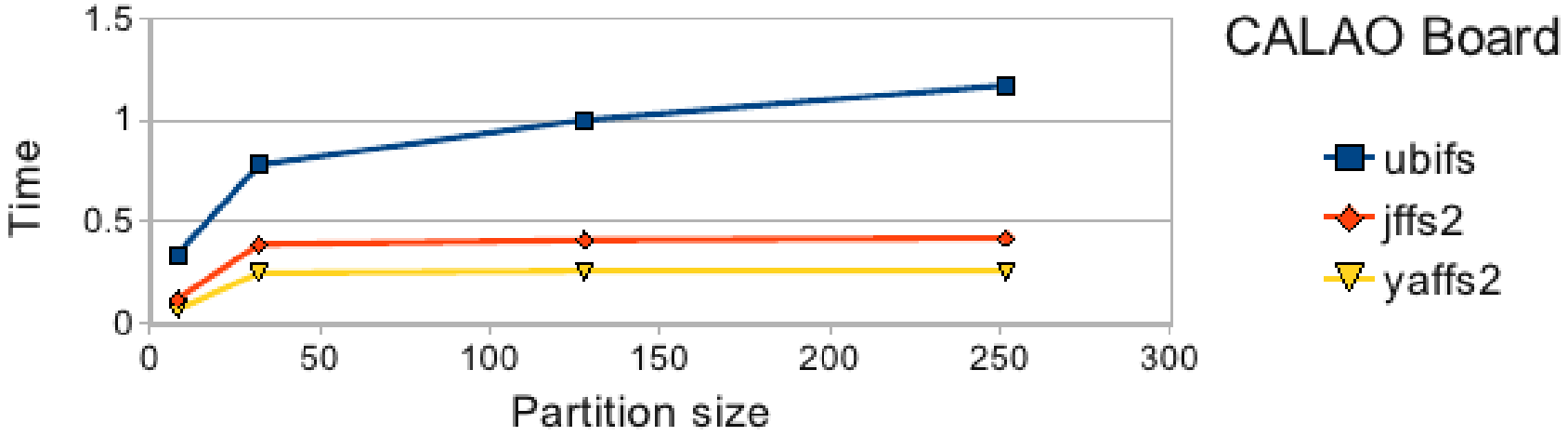
UBIFS

- UBIFS zu nutzen ist komplexer
  - UBI Image erzeugen
    - mkfs.ubifs
  - Image flashen
    - UBI Container erzeugen
    - UBI Volume erstellen
    - Volume mit Image füllen
  - UBIFS mounten
    - MTD Device mit UBI Layer verbinden
    - UBI Volume mounten

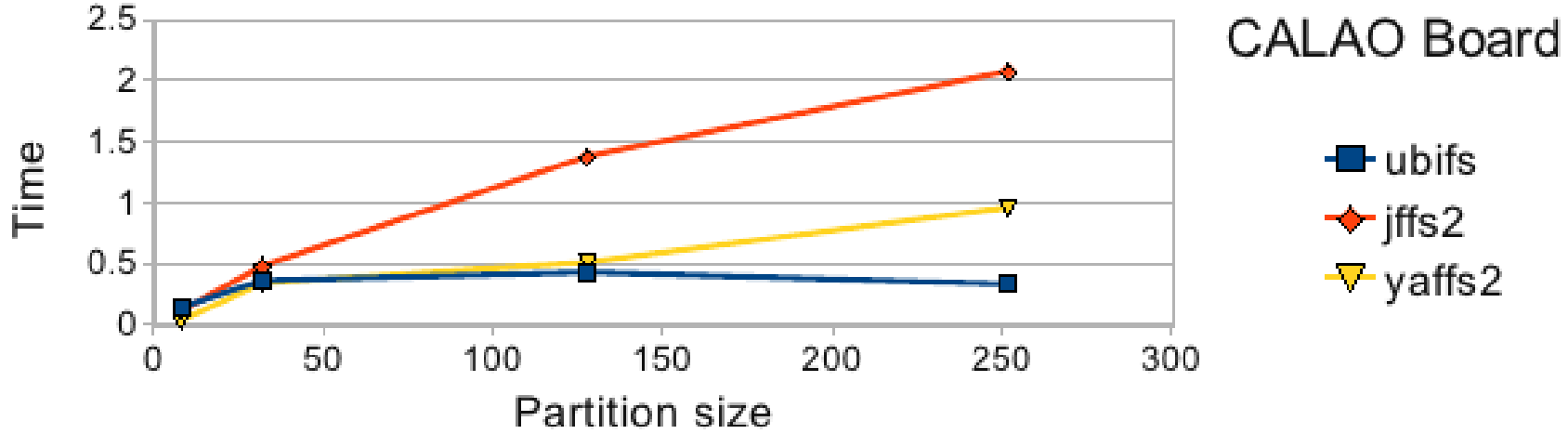


Flash Filesysteme Benchmarks

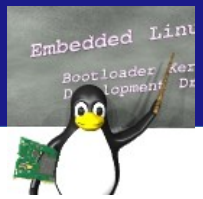
Initialisierung



Mounten

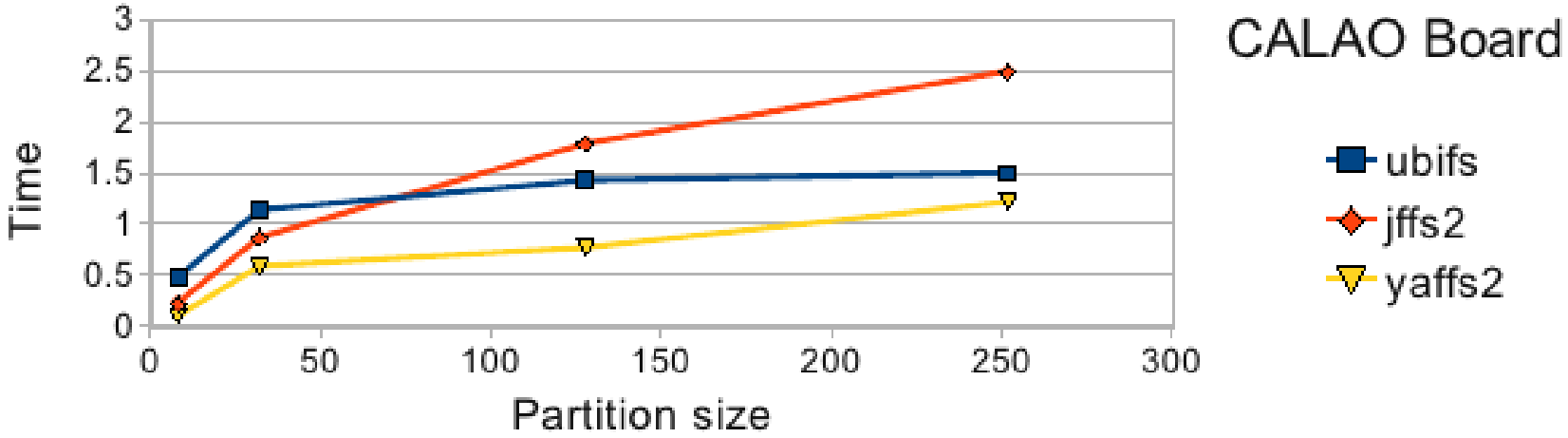


aus Flash filesystems benchmarks, Free Electrons, 2010

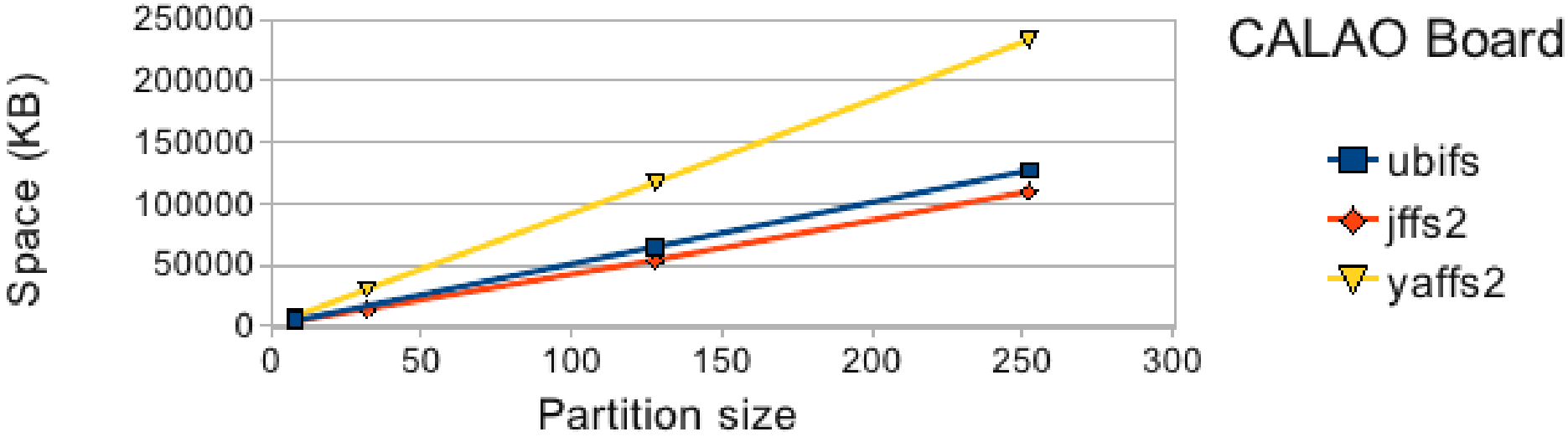


Flash Filesysteme Benchmarks

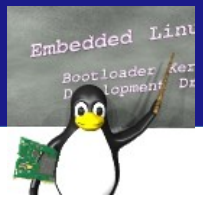
Initialisierung + Mounten



Speicherverbrauch

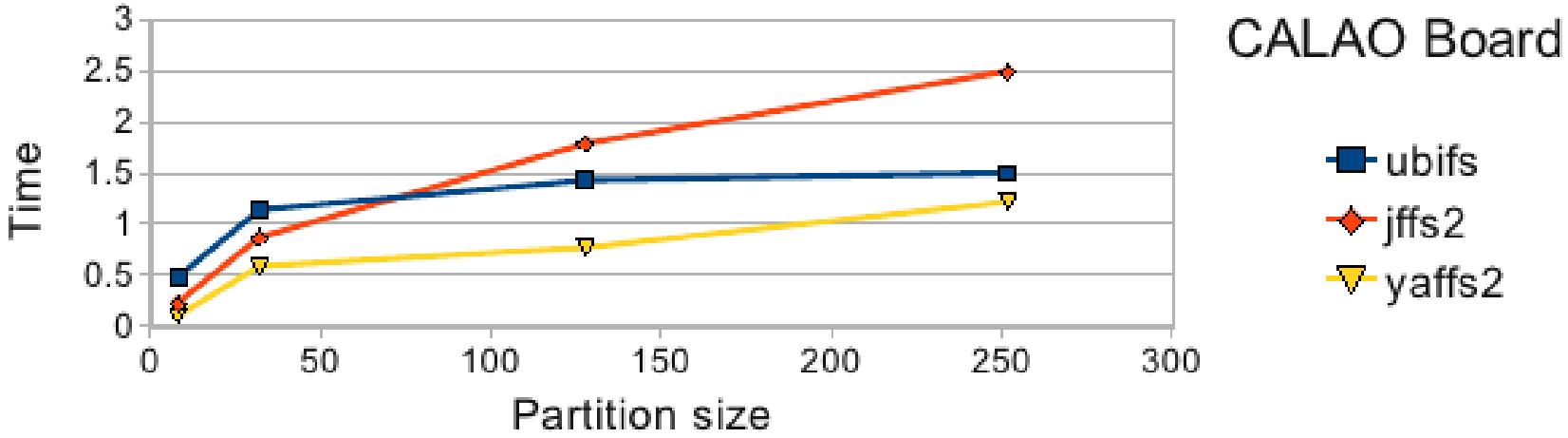


aus Flash filesystems benchmarks, Free Electrons, 2010

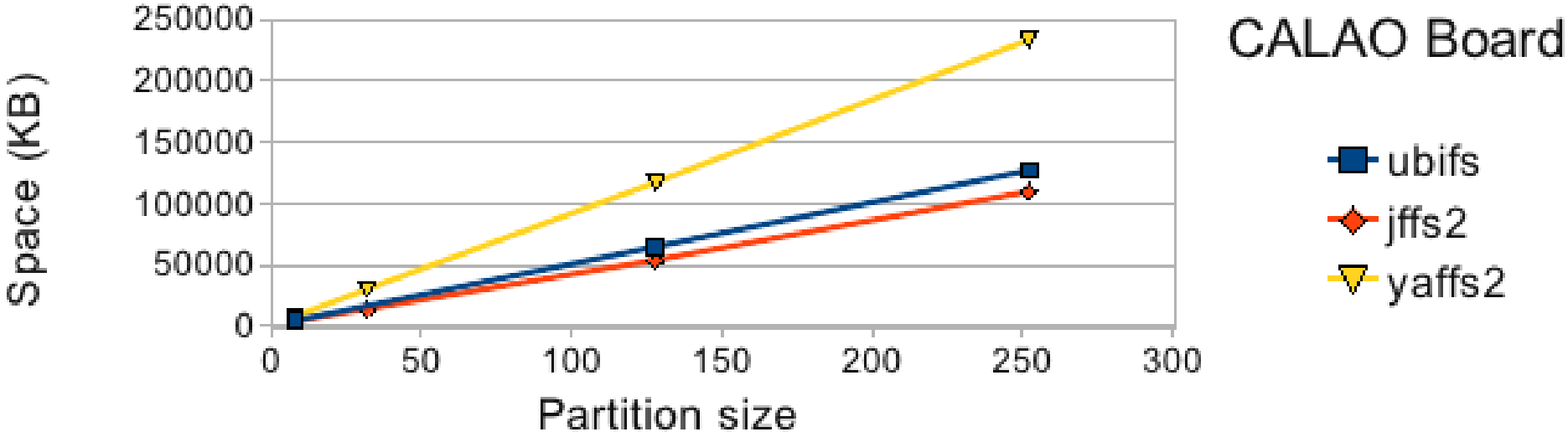


Flash Filesysteme Benchmarks

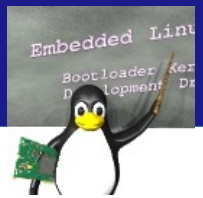
Initialisierung + Mounten



Speicherverbrauch

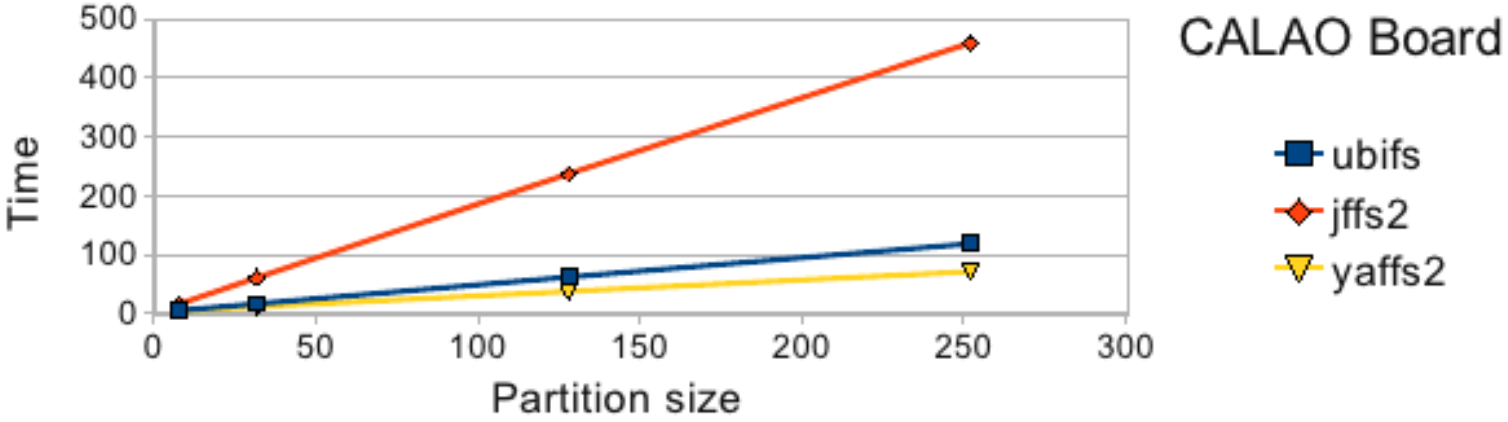


aus Flash filesystems benchmarks, Free Electrons, 2010

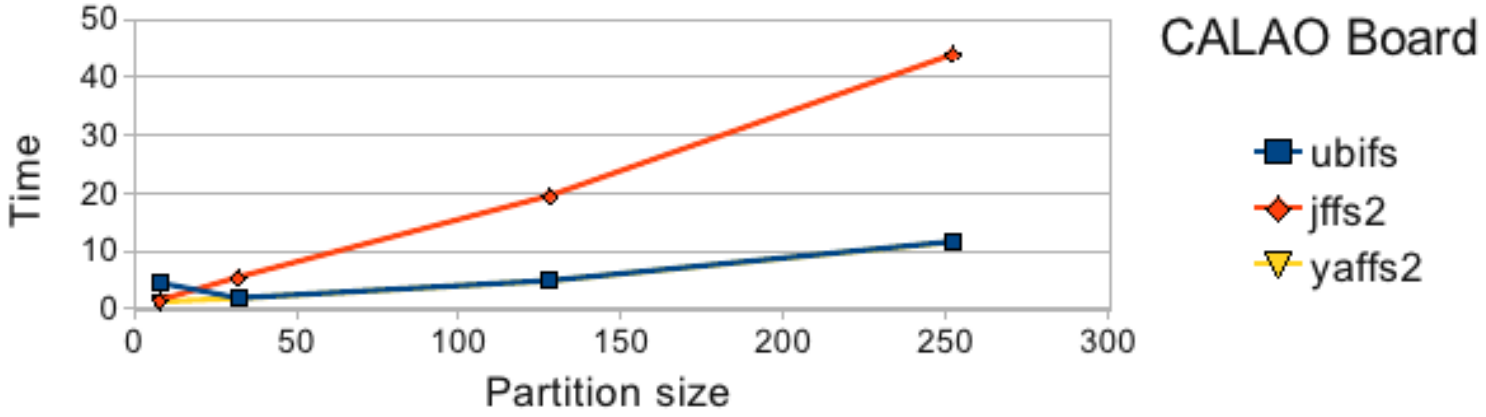


Flash Filesysteme Benchmarks

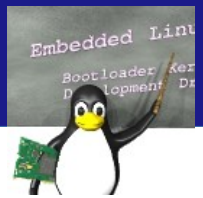
Lesen



Schreiben

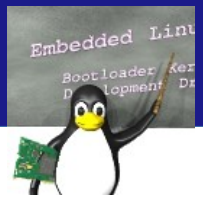


aus Flash filesystems benchmarks, Free Electrons, 2010



EXT2/3/4

- Extended File System ext
  - Als Ersatz für minix für Linux 1992 entwickelt
  - Maximal 2 GB Partitions-Größe
  - Maximale Länge der Einträge 255 Zeichen
- ext2
  - 1993 released
  - Maximal 4 TB Partitions-Größe
- ext3
  - ext2 erweitert um Journal
  - ext3 lässt sich als ext2 mounten



EXT 2/3/4

- Partition ist in logische Blöcke aufgeteilt
  - Die Größe der Blöcke kann beim Formatieren festgelegt werden Vorgabe 4kB
  - Die Blockgröße beeinflusst die Performance
    - Kleine Blöcke: Bessere Ausnutzung der Partition bei kleineren Dateien. Mehr Verwaltungsaufwand, langsamerer Zugriff.
    - Große Blöcke: Mehr ungenutzter Speicher. Weniger Verwaltungsaufwand, schnellerer Zugriff.