Schnelleres Suspend-to-NVRAM in PAVE/FreeBSD

Masterprojekt

25.03.2025

Jonathan Krebs

Friedrich-Alexander-Universität Erlangen-Nürnberg

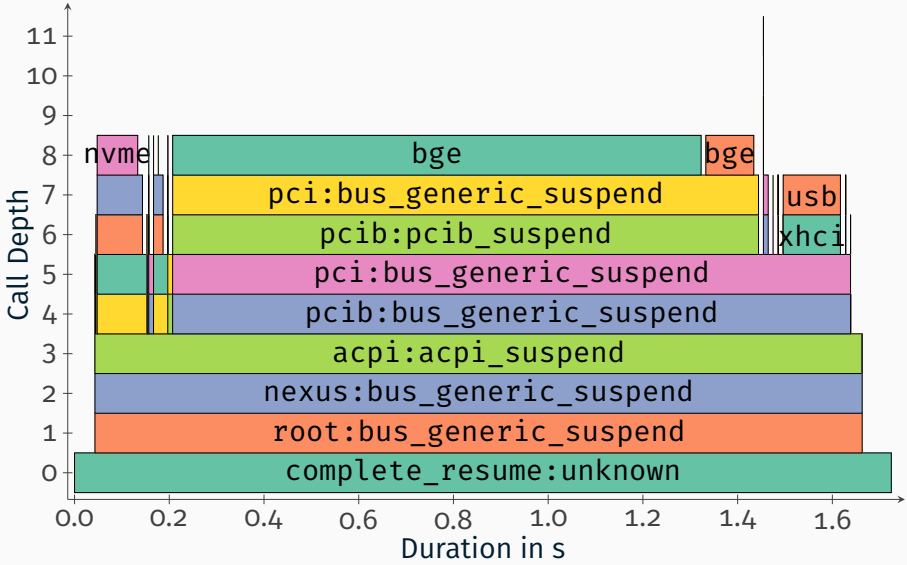
Lehrstuhl für Informatik 4
Systemsoftware

Friedrich-Alexander-Universität
Technische Fakultät

Motivation

- In PAVE soll FreeBSD nach Stromausfall transparent weiterlaufen
- System läuft in nicht flüchtigem Hauptspeicher (NVRAM)
- Bei Stromausfall flüchtigen Zustand mit Restenergie sichern
- Aktuell: Suspend-to-NVRAM analog zu Suspend-to-RAM
- Suspend der Geräte macht Großteil der Zeit (1.6s, ca. 95%) aus

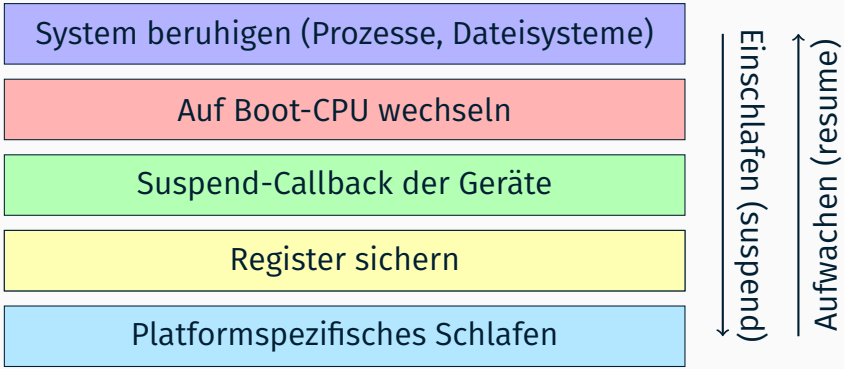
Ausgangszustand



Anforderungen

- Suspend-Dauer reduzieren
- Treiberunabhängig
- Transparent für Anwender

Suspend-to-(NV)RAM in FreeBSD

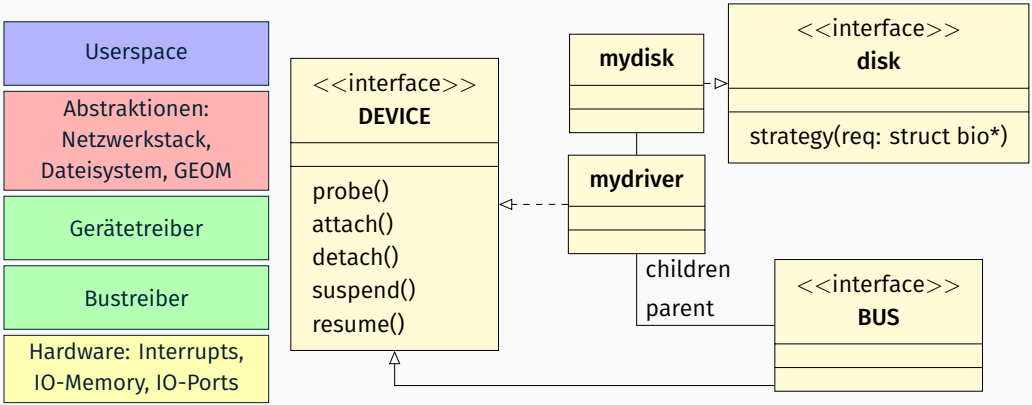


Gerätetreiber: Busse und Hierarchie

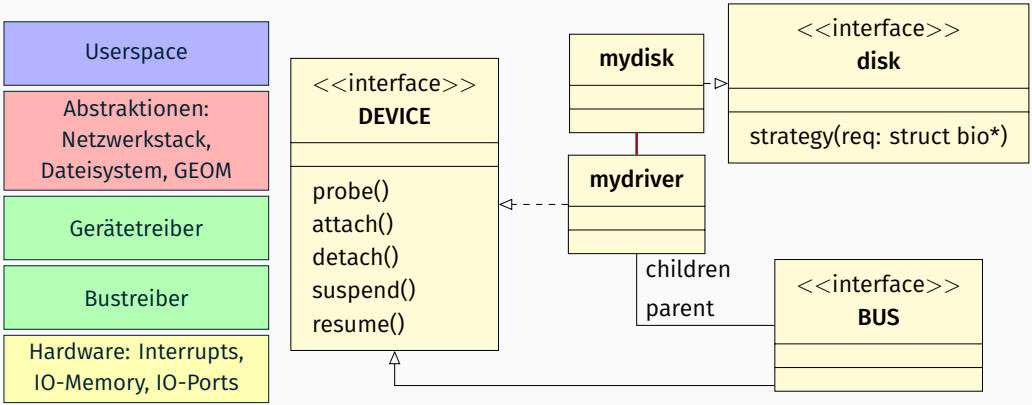
- Geräte bilden einen Baum
- Innere Knoten heißen Busse,
  - Hardware-Busse wie z.B. PCI, USB,
  - Hierarchie durch Software (`root_bus`, `nexus`, ACPI)
  - stellen Bus-Spezifische Hilfsmethoden zur Verfügung (PCI, virtio)
- In ACPI/NVRAM-Suspend-Routine: `DEVICE_SUSPEND(root_bus);`

```
jonny@charon-freebsd:~$ devinfo
nexus0
  efirtc0
  smbios0
  ram0
  apic0
  acpi0
    acpi_ec0
    cpu0
      acpi_perf0
      hwpstate_intel0
      cpufreq0
    ...
    pcib0
      pci0
        vgapci0
        xhci0
          usbus0
            uhub0
              umass0
        pcib4
          pci4
            nvme0
        isab0
          isa0
        em0
```

Gerätetreiber: Schnittstellen



Gerätetreiber: Schnittstellen



Problem

Ober- und Unterseite eines Treibers lassen sich von außen nicht zuordnen

Ansatz: Unterseite

- Suspend: nichts tun
- Resume: Gerät wiederbeleben.
  - Kein `resume()` ohne `suspend()`
  - `suspend()` und `detach()` erwartet bekannten Gerätezustand. Nicht nach dem Reboot möglich
  - PCIE Hot-Unplug:
    - Signalisiert durch Interrupt und IO-Memory konstant oxFFFFFFFF
    - IO-Memory emulieren
    - `detach()`
    - IO-Memory wieder herstellen
    - `attach()`

Folge für Oberseite

Flackern: Abstraktionen der Geräte verschwinden und kommen wieder.

Ansatz: Oberseite

- Flackern verstecken: Geräte (Einträge in `/dev/` und `ifconfig`) virtualisieren und Aufträge / Konfiguration puffern
  - Aufwändig: muss pro Gerätetyp implementiert werden
  - benötigt Zuordnung der Treiberoberseiten vor und nach dem Resume
  - Wenn nur grundlegend implementiert, droht Geschwindigkeitsverlust (Netzwerkkarte hat z.B. nur noch eine Queue, keinerlei Offloading)
- Flackern sichtbar für Benutzer (Userspace, Dateisystem)
  - Programme / Netzwerk robust konfigurieren :)
  - Dateisysteme unglücklich bei Verschwinden des Speichermediums (Kernel-Panic / EIO)
- Implementiert: Mischform. Geräte flackern, aber Speichermedien durch GEOM-Ebene stabilisiert

Implementierung: Speichermedien stabilisieren

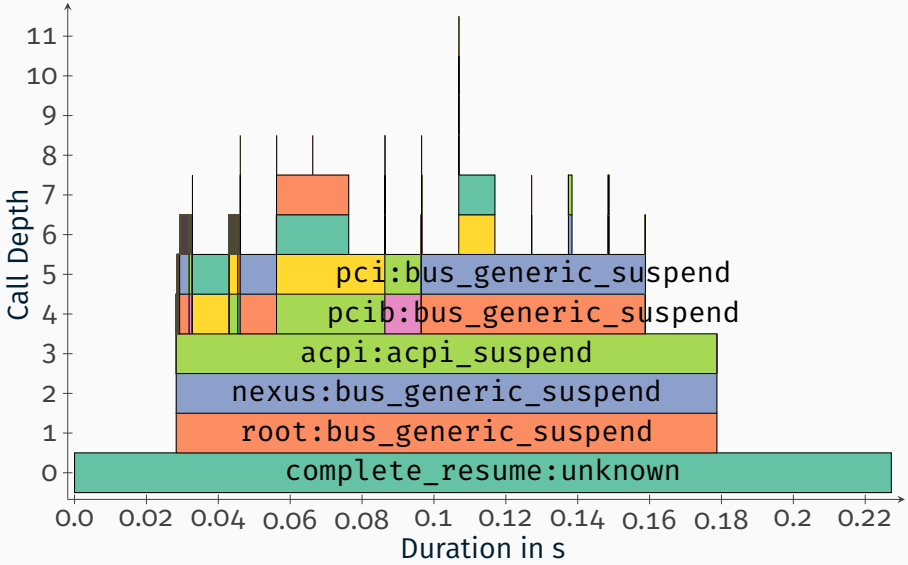
Probleme

- Verschwinden führt zu Unmount, teilweise Kernel-Panic
- für Konsistenz: suspend wartete auf Abschließen von Aufträgen.

Lösung

- unterbrochene Aufträge nach dem Aufwachen wiederholen
- Schreibpuffer in Festplatten beachten: Schreibauftrag erst nach „Flush"' abgeschlossen
- Implementiert auf Basis des vorhandenen GEOM-Moduls `g_mountver`. Neu:
  - Initialisierung beim Booten,
  - Warten auf und Veranlassung von Flushes

Evaluation: Zeitverbrauch beim Suspend



Evaluation: Erreichte Ziele

- Suspend-Dauer reduziert: drastisch reduziert. Verbleibend: 227ms
- transparent für Anwender: bei richtiger Konfiguration
- treiberunabhängig
- konfigurierbar, welche Geräte betroffen sind

Future Work: Weitere Optimierungsmöglichkeiten

- Stromausfall-Suspend ist greifbar, ausprobieren
- Wie viel von der Systemberuhigung ist wirklich notwendig?
- Lock um das Suspend und Resume der Geräte kann zu undefiniert langer Verzögerung führen

Ende

Fragen / Diskussion

Implementierung: Suspend-Callbacks unterbinden

Anforderung: Es soll für jedes Gerät festgelegt werden können, ob `suspend( )` übersprungen werden soll.

(a) Patch in `bus_generic_suspend`

- Standardimplementierung von `suspend( )` in vielen Bussen oder wird dort aufgerufen, um rekursiv `suspend( )` der Kinder aufzurufen
- breite Abdeckung, aber ein Bus-Treiber könnte die Iteration selber implementieren
- jedes Gerät (`struct _device`) bekommt Attribut `bool nosuspend`: Suspend-Callback überspringen?
- Attribut durch `sysctl` konfigurierbar

Implementierung: Suspend-Callbacks unterbinden

Betrachtete Alternativen:

(b) Als Knoten im Gerätebaum
- Konzeptionell einfach, keine tiefgreifenden Änderungen nötig (als Modul implementierbar)
- Aber: 500 Zeilen Boilerplate zum Durchreichen der PCI- und Bus-Funktionen, Bus-Spezifisch

(c) Quellcode der Treiber anpassen:
- nicht implementiert, da nicht treiberunabhängig

(d) V-Table der Objekte zur Laufzeit patchen:
- keine tiefgreifende Änderung im Kernel notwendig
- aber dennoch implementierungsspezifisch
- nicht implementiert.

Implementierung: Resume

```
def nosuspend_resume(d):
 m = remap_io_memory(d) # PCIE specific
 d.detach()
 undo(m)
 d.probe()
 d.attach()

def remap_io_memory(d):
 m = []
 for r in d.parent.get_resource_list(d):
  if r is memory and is mapped:
   for page in r:
    save virtual address and page table entry of page to m
    allocate, map and memset a new page
 return m
```