Linux Suspend-To-Disk Objectives for Consumer Electronic Devices

Vitaly Wool
EmbeddedAlley Solutions Inc.

Suspend-To-Disk (STD)

- suspend mode with the most power saving
- may be deployed on systems with minimal PM
- writes all the memory contents to disk before going into suspend
  - swap or dedicated partition
- the kernel is re-started on resume
  - restores the information from STD image

STD VS STR

- data saved to non-volatile storage
- userspace data is saved
- resume at kernel entry point
- devices are first put into active state, then suspended/resumed

- data saved to RAM
- no userspace data is saved
- resume at kernel resume point
- devices are resumed directly

STD Objectives

- Pro
  - maximum power savings
  - easy to implement for a new platform
- Contra
  - long time to suspend
  - long time to resume
  - needs a lot of free space on disk
  - redundant data copying
    - storage to allocated space, then to RAM

STD for a CE device

- CE device objectives
  - usually no disk, only flash
  - struggle for space saving
  - slower than PC
  - usually powered even in OFF state (battery)
- “vanilla” STD not suitable
  - see the previous slide
    - the resume time is not just long, it's VERY long

STD for a CE device: how to adapt?

- boot time
  - more support from bootloader
  - parallel device init (separate topic :))
  - skip kernel init stage
- space consumption
  - selectively save the information
  - compress the data

STD example: swsusp for ARM

- data saved on a swap partition
  - MTD usually
  - flushing takes quite a bit of time
- suspend using the standard means
  - 'echo disk > /sys/power/state'
- resume gets a bit of bootloader help
  - “resume=/dev/mtdblock2” or such

STD example: suspend2 for ARM

- data saved on a swap partition
  - MTD usually
  - flushing takes quite a bit of time
- compression and encryption available
- suspend using the standard means
  - “hibernate” script
- resume gets a bit of bootloader help
  - “resume2=/dev/mtdblock2” or similar

STD example: snapshot boot for ARM

- data saved on a swap partition
  - MTD usually
  - flushing takes quite a bit of time
- suspend using the standard means
  - 'echo disk > /sys/power/state'
- resume needs bootloader help
  - 'bootss <address>' command

snapshot boot VS STD

- direct data copying
  - swap to RAM
- starts at kernel resume point
- devices are resumed directly

- redundant data copying
  - swap to allocated
  - allocated to original
- starts at kernel entry point
- devices are first put into active state, then suspended/resumed

Example: PocketPC prototype

- ARM1136-based SoC
- supports Deep Idle state
  - the CPU is turned off, only RTC clock is running
  - most of the registers are lost
  - resumes on power button press, RTC alarm and «battery low» signal
  - RAM self-refresh mode available
- battery always in
  - RAM contents are not lost on exit from self refresh

Example: PocketPC prototype

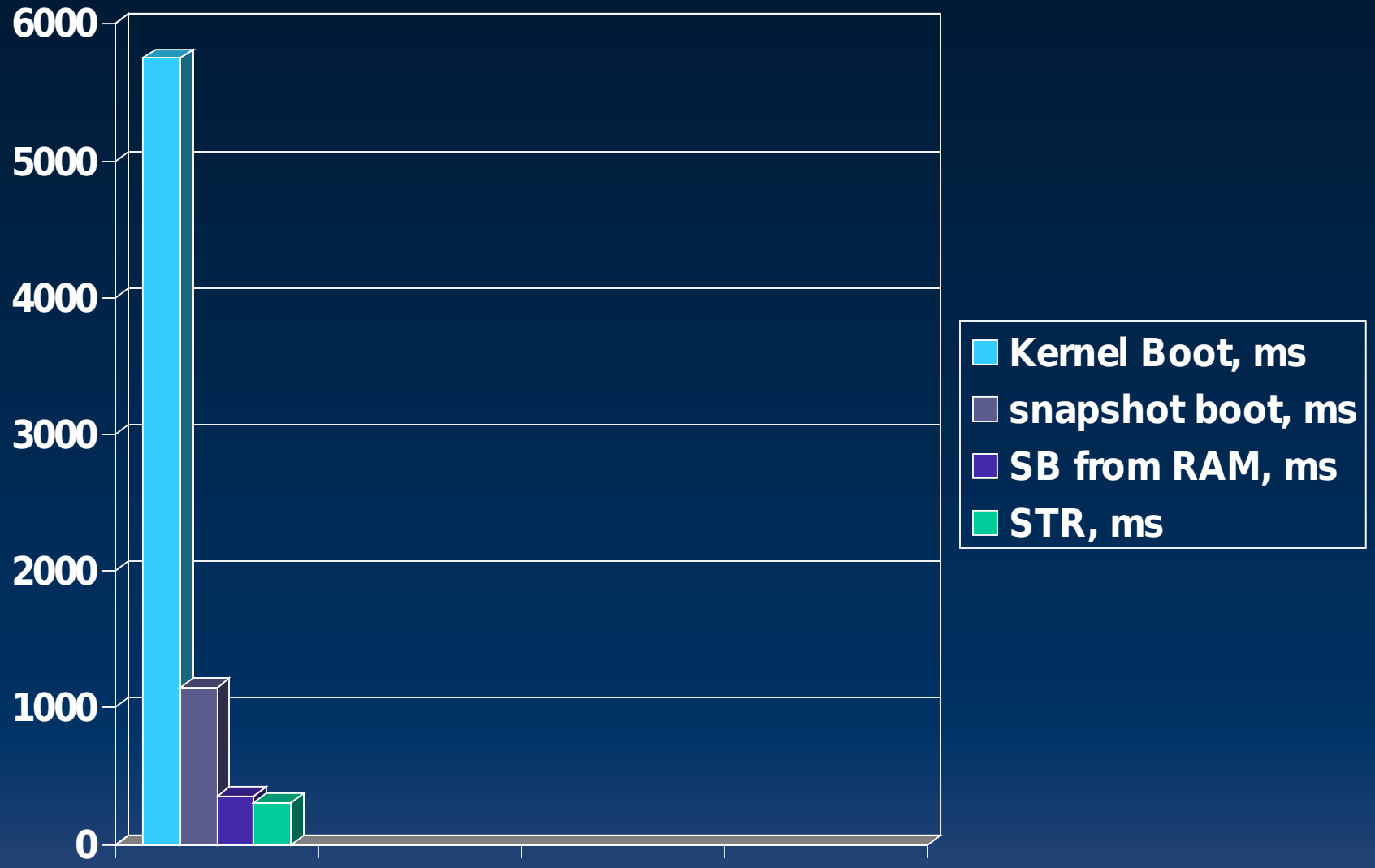
- power management goals
  - aggressive power saving strategy
  - fast resume
  - state preservation if the battery is low
- STR?
  - fails the last goal
  - a lot of work to preserve CPU and devices' registers/states
- STD?
  - fails the second goal

Solution: “snapshot boot + STR” hybrid

- modify “snapshot boot” to cleanly separate saved kernel and userspace info
- use “snapshot boot” for suspend but save data in RAM
- wakeup on “battery low” event and flush data to flash or disk

“snapshot boot + STR” hybrid: how it works

- resume starts running bootloader either way
- bootloader to find out
  - normal boot
  - resume from Deep Sleep (data in RAM)
    - restore only kernel-related data
  - resume from poweroff (data in swap)
    - standard resume from snapshot boot


6000
5000
4000
3000
2000
1000
0
Kernel Boot, ms
snapshot boot, ms
SB from RAM, ms
STR, ms

Conclusions

- current Linux suspend states do not always match the CE devices' PM goals
  - more flexibility is desired
- hybrid solutions can help in many cases
  - better matching the goals
  - existing code/framework reuse
- what about standardizing?
  - ...

Questions?

mailto:vital@embeddedalley.com