Improving Linux Startup Time
Using Software Resume

Hiroki Kaminaga
kaminaga (at) sm.sony.co.jp
\$ who am i

- Work for Sony Corporation
  - Corporate Advanced Technology Development
  - Provide Linux to product development teams
    - TV
    - Mobile (battery powered device)
    - Video Recorder
    - Etc.
The Problem of System Startup Time

- System startup done before application is ready
  - Transfer kernel image and userspace pages to RAM
  - Linking dynamic shared library
  - C++ global constructor execution
  - Application startup time IPC

- Application startup time dominates system startup time

<table>
<thead>
<tr>
<th>HW Init (Boot loader / kernel)</th>
<th>Mount driver init</th>
<th>Application startup time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 [s]</td>
<td>2 [s]</td>
<td>5 [s]</td>
</tr>
</tbody>
</table>
Existing Methods of Reducing Startup Time

• Prelinking
  – Pre-calculate dynamic relocation
  – Reduces dynamic link processing time
  – Only fixes one factor, and increases binary size

• XIP (eXecute In Place)
  – Executes directly from NOR flash or ROM
  – Reduces the copy of kernel and userspace pages
  – Slow execution, limited by system design
The Need for Snapshot Boot

• Existing methods only address individual parts of the problem, and do not provide a complete solution

• The proposed snapshot boot method tries to provide a comprehensive solution to the startup time problem

• Configuration rarely changes in embedded systems.
What is Snapshot Boot

• Snapshot boot uses the existing system resume to reduce application startup time by loading a fixed, pre-made system image to RAM.

• Snapshot boot uses boot loader/kernel cooperation to minimize the system initialization needed prior to resume

• Snapshot boot provides a comprehensive solution
Why Utilize Resume

*Normal Start up time*
- copy text/data to RAM
- I/O init
- run time state init
  - IPC sync/wait/sched
  - Dynamic link
  - Global Constructor execs
  - Other runtime init.

*Ideal Resume time*
- copy text/data to RAM
- I/O init
- transfer state to RAM
  - Heap
  - Stack

• Typically (run time state init) >> (transfer state to RAM)
Preparations

• Before the details of Snapshot Boot…
• Introduction to software suspend/resume (swsusp)
  – Already in 2.6 kernel
  – Documentation/power/swsusp.txt
Suspend Methods in Linux

- Standby
- Suspend to RAM
- **Suspend to Disk**
  - writes runtime state to system image on non-volatile storage.
  - resumes pre-suspended state from system image
Suspend to Disk Demo
Target Environment

- Target board
  - OMAP Starter Kit (OSK5912)
- Boot loader
  - U-boot 1.1.4
- OS
  - Linux 2.6.11
- Application
  - mplayer

Hardware Features:
- ARM9 core operating at 192 Mhz.
- DSP core operating at 192 Mhz.
- TLV320AIC23 Stereo Codec
- 32 Mbyte DDR SDRAM
- 32 Mbyte Flash
- RS-232 Serial Port
- 10 MBPS Ethernet port
...
Target Adaptation

- Flash ROM is used as non-volatile storage
- Porting to ARM was needed
Details of Suspend to Disk

- `freeze_processes()`
- `free_some_memory()`
- `device_suspend()`
- `device_power_down()`
- `save_processor_state()`
- Make snapshot image (in memory)
- `device_power_up()`
- `device_resume()`
- `write_suspend_image()`
- Powerdown machine
Details of Resume From Disk

- `software_resume()` called at `late_initcall`
- Read system image from storage to RAM
- `freeze_processes()`
- `device_suspend()`
- `device_power_down()`
- Restore system image
- `restore_processor_state()`
- `device_power_up()`
- `device_resume()`
- `thaw_processes()`
Why Resume From Disk Takes So Long

• Resume processing is done after almost all normal kernel startup is done
  – Devices used for resume startup are then shutdown
  – Devices are then re-initialized by the resume code

• System image copy is done twice
  – storage to working buffer
  – working buffer to final address

• Process freeze takes time
Snapshot Boot: Boot Loader Detail

• Boot loader and kernel cooperate
• Procedure on boot loader side:
  – *Wakeup board*
  – Copy system image to RAM
  – Minimal device setup for resume
  – Jump to kernel resume point

Usual procedure

Added procedures
Snapshot boot: Kernel Detail

- `software_resume()` called at `late_initcall`
- Read system image from storage to RAM
- `freeze_processes()`
- `device_suspend()`
- `device_power_down()`
- Restore system image
  - `restore_processor_state()`
  - `device_power_up()`
  - `device_resume()`
  - `thaw_processes()`
Snapshot Boot: Flow Diagrams

Resume from disk flow diagram:

Kernel Land

- Load kernel dev
- Init dev
- Setup
- Copy image
- Restore resume dev
- Continue process

Boot Loader

- Init board

ON

Snapshot boot flow diagram:

Kernel Land

- Restore resume dev
- Continue process

Boot Loader

- Init board
- Copy image
- Setup dev

ON
Snapshot Boot: Demo
Snapshot Boot: Implementation

- Boot loader support implemented as new command in u-boot
  - bootss <image addr>

- Boot loader startup sequence:
  - Setup clock speed, timer, MMIO regs
  - Copy system image from flash to RAM
  - Setup MMU
  - Jump to kernel resume point

- Minor kernel modification
  - Added resume entry point, enabled interrupts
.MAP5912 OSK  # reset
U-Boot 1.1.4 (Jul 6 2006 - 16:57:24)
U-Boot code: 18888880  -> 18897A7B  BSS:  -> 1889C27C
RAM Configuration:
Bank #0: 18888880 32 MB
Flash: 32 MB
In:  serial
Out: serial
Err: serial
Hit any key to stop autoboot: 0
cmd: bootss, argc: 2, at: 0x10007f00
argv[0]: bootss
argv[1]: 00240000
   sizeof(swsusp_header): 4096 (0x00001000)
   sizeof(swsusp_info): 3408 (0x00000da0)
bootss: changed to clock to 152MHz
bootss: OMAP MPU timers initialized
   pages to copy: 2354 (0x0000932)
bootss: copy image done.
bootss: jumping to kernel resume point: 0xc0154c04
[ 4.094449] Restarting tasks... done
U: 10.7 556 2012.251% 0.8% 0 0 0 0
CTRL-C  For help 115200 BNT : NODE : Minicom 2.0.0 : UT102 : Offline
Some Issues Faced

• Many drivers don’t properly implement resume, and just rely on the initialization done at startup.
  – Current snapshot boot implements device initialization at boot loader as a workaround.
  – Similar issue in kexec too…

• System image format changed in recent kernel
Conclusion

• Successfully implemented snapshot boot feature
• Reduced system startup time by 50%
• Implemented Suspend to Disk for ARM
More Information

- Suspend to Disk and snapshot boot for ARM wiki page is available at CE Linux Forum website
  http://tree.celinuxforum.org/CelfPubWiki/SuspendToDiskForARM
- Target board wiki page
  http://tree.celinuxforum.org/CelfPubWiki/OSK
- Come to CELF Project BOF meeting Fri, 7pm, at Les Suites