TATA

TATA ELXSI

engineering creativity

Boot time Optimization of Automotive Grade Linux

Shilu SL & Renjith G

14-Jul-2016

Agenda

- Importance of Fast Boot in AGL
- Setting up of targets
- Boot time optimization techniques
- Explaining with a live example
- Conclusion
- Q&A

Why Fast Boot for AGL

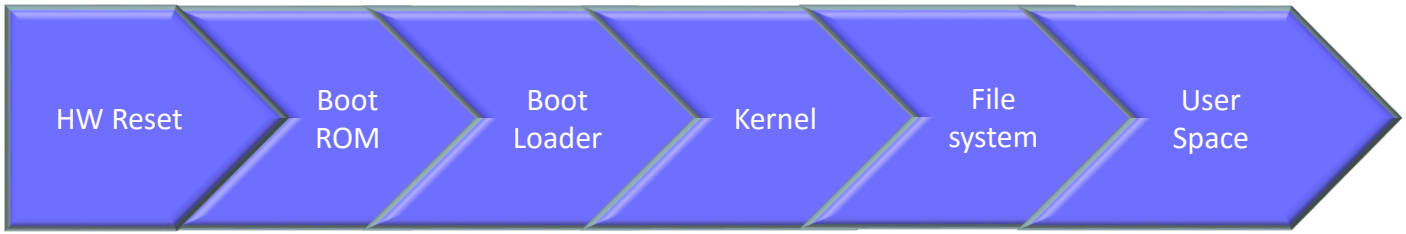
- Differentiate AGL from main stream Linux
- Making AGL compete with any other traditional RTOSs
- Enhancing user experience
- Legal requirements
- Power saving (reducing standby power)
- Increasing SoC complexity and Boot time
- Increasing application complexity
- Issues with Hibernation, Snapshot boot and Sleep methods
- Marketing

Setting the targets

- End application and use case
- Required functionalities
- Required flexibilities
- Hardware limitations
- Commercial needs and limitations

AGL Booting Sequence

- Same as mainstream Linux
- Difference in use cases
- Limitations in HW selection



HW Optimization

- SoCs designed for automotive applications are limited
- The SoC architecture and the board design are influenced by many other factors and most of the time the fast boot requirements were not taken care of but it is an important thing affecting the boot time.
- Possible optimizations
  - Quick HW reset logic
  - Good power supply design
  - Good boot logic
  - Faster boot media
  - Careful selection of peripherals (with fast initialization possible)
  - Faster memory

Boot ROM Optimization

- Not much optimization scope as it is too simple
- Most of the time this will be supplied by the SoC vendor with little chance of customization
- Being from the SoC vendor this is expected to be fully optimized but it is not the case most of the time
- Can consider single step boot loader

Boot Loader Optimization

- Tasks
  - Basic setup of CPU like setting up clock, memory
  - Preparing and handing over device trees
  - Clean up like flushing the cache
  - Relocate Linux Kernel from Flash to RAM (most time consuming)
  - Handover some parameter and switch to Linux Kernel
- Optimization Possibilities
  - Remove unwanted/unimportant features/tasks
  - Reduce the size of Linux Kernel

Kernel Optimization

- Lots of optimization scope
- We can select the configuration
- We can select the compression type
- Adjust boot parameters
- Initialization
  - Waiting for device driver initialization is most time consuming
  - Can differ
  - Make it parallel where ever possible

File System Optimization

- Mounting file system is time consuming
- UbiFS is recommended to achieve faster boot if using flash devices
- Take the transition table and write it to the flash
- InitRAMFS
- Customize services

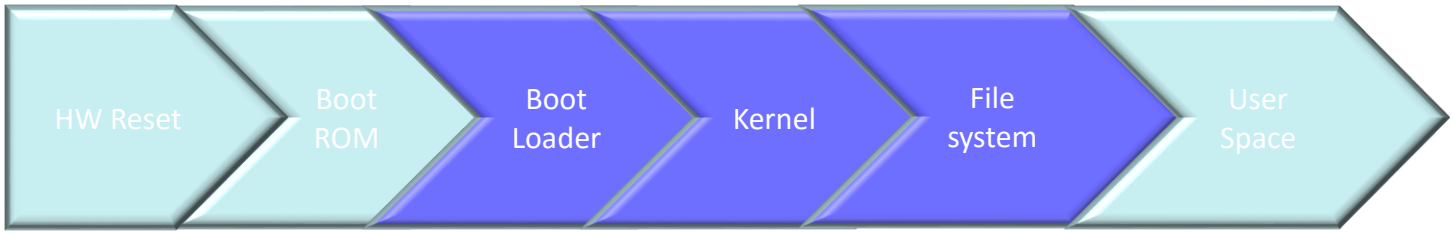
Application Optimization

- Loading from storage device is time consuming task
- Lots of optimization scope
- Combine init scripts
- Remove sanity checks (if confident)
- Move unnecessary items to a script and start it later

Live Example

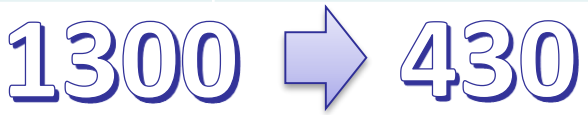
Component	Version / Type
U- Boot	2013.1
Kernel	3.10.31 LTSI
File system	EXT4 (AGL)
Platform	R-Car M2 (Koelsch)
Build Environment	Yocto Project 1.7 (Dizzy)

- Target = Reaching user space in < 2.5 seconds
- Constraints = No hardware and Boot ROM modifications
- Simple application rendering 2D images on a HDMI monitor



U-Boot Optimization

No	Change	Time reduction
1	Bug fixes	20ms
2	Boot appended DTB image	50ms
3	Remove CRC Verification	30ms
4	Disable MEMTEST	50ms
5	MMC and SD pre-initialization	40ms
6	Reorganize MMC and SDHI initialization	40ms
7	Move eth pins initialization to board_eth_init()	50ms
8	Add DMA driver and QSPI DMA support	350ms
9	Load time reduction based on kernel size	140ms
10	Disable some of the prints	100ms
	Total time reduction	**870ms**

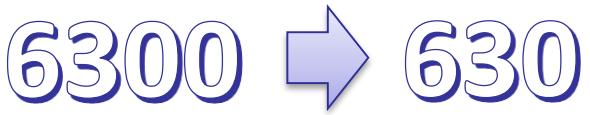


Kernel Optimization

No	Change	Time reduction
1	Remove unnecessary modules	3100ms
2	Disable initilaization of serial port1, SDHI1, SDHI2 & SATA	200ms
3	Adjust bootargs	300ms
4	Disable console output	1000ms
5	Initialize DMA early	40ms
6	Disable SDIO/MMC scan	30ms
7	Initialize high memory in separate thread	150ms
8	Reduce mount delay	80ms
9	Remove Trace support	30ms
	Total time reduction	**4930 ms**

Filesystem Optimization

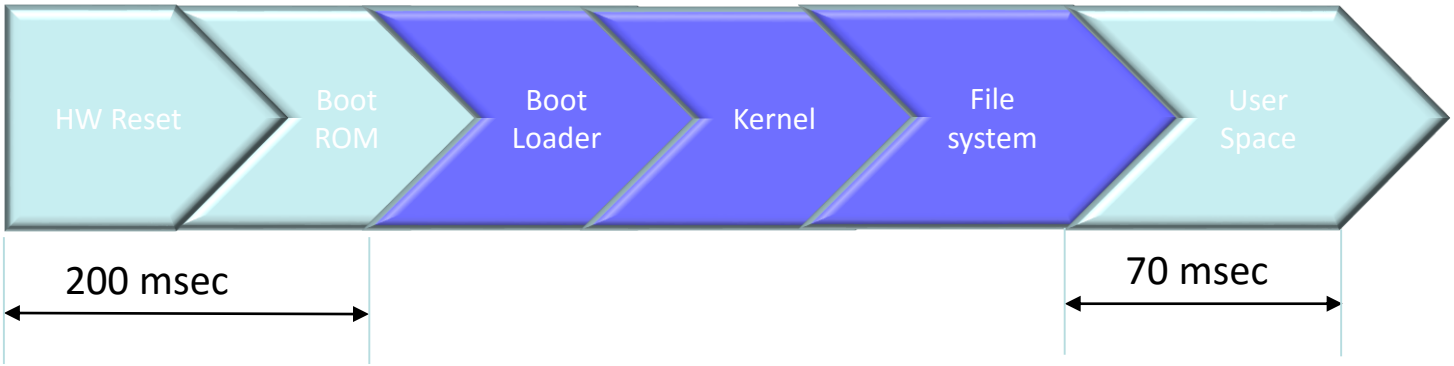
No	Change	Time reduction
1	With ext2/ext3/squashfs	No Gain
2	Auto login	1000ms
3	Disable services	3300ms
4	Disable console output	1000ms
5	Disable ext4 journaling	100ms
6	use fbdev-backend instead of drm and disable pvr	170ms
7	Optimize udev rules	100ms
	Total time reduction	**5670 ms**



Conclusion

Achieved the target by bring down the boot time from 13.6 sec to 2.13 sec

13600 ⇨ 2130 (84%)



Demo

Q&A

Thank you

TATA ELXSI

ITPB Road, Whitefield
Bangalore, India 560 048
Tel +91 80 22979123
Fax +91 80 28411474
e-mail info@tataelxsi.com

www.tataelxsi.com