Tips and techniques for improving embedded Linux startup time

Guests

- Sridharan Subramanian, Software & Platforms Prod Mgmt Lead
  - Freescale Semiconductor
- Christopher Hallinan, Field Applications Engineer
  - MontaVista Software

Presented by

Moderator: Don Dingee, OpenSystems Media

Agenda

- A bit of housekeeping
- Moderator thoughts on topic
- Expert guest presentations
- Q&A – send us your questions

Face it, we're impatient

- How long is too long to wait for a boot?
- Recent blog from the other OS camp: "… a very good system is one that boots in 15 seconds …"
- Nowhere near good enough in many embedded apps
- It depends what OS, and how much you really need

Not just consumers need 'instant-on'

- **Smartphones have set the bar**
- **Infotainment devices are following – in the car and home**
- **Medical devices need speed**
- **Industrial devices too**
- **As expectations drop, shaving seconds counts more**
- **And Linux can get the job done**

Question/Answer

Guests

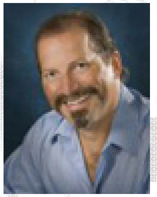
- **Sridharan Subramanian, Software & Platforms Prod Mgmt Lead**
  - **Freescale Semiconductor**
- **Christopher Hallinan, Field Applications Engineer**
  - **MontaVista Software**

freescale™ semiconductor

montavista™

Tips and techniques for improving embedded Linux startup time

Presented by MontaVista Software and Freescale Semiconductor



Chris Hallinan
Field Applications Engineer
MontaVista Software

Sridharan Subramanian
Software & Platforms Product Manager
Freescale Semiconductor

montavista™

freescale™
semiconductor

Contents

- ► Challenges facing consumer product design
- ► Embedded hardware and software platforms to address these challenges
- ► Software techniques to decrease boot time
- ► Overview of boot sequence
- ► Linux kernel optimizations

Multimedia Devices – Explosive Growth

▶ Cellular

- 4 billion subscribers by the end of 2010

▶ Mobile Multimedia

- >200M Portable Media Players sold/year by 2010
- >50M Personal Navigation Devices by 2010

▶ Auto Infotainment

- 60% of worldwide vehicles Bluetooth enabled by 2013 at a 40% CAGR.

▶ Consistent Trends

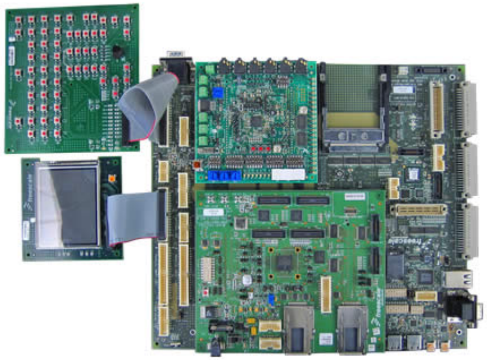
- Video becoming standard in PMP
- GPS proliferation in cellular and auto
- Connected devices – 3G, Bluetooth, Wi-Fi
- Richer UI with browser

Sources: Cowan, In-stat, iSuppli

What do consumers want?

- ‘Cool’ devices
- Enhanced features – Video and connectivity
- Quality
- Performance and battery life
- Instant-on; Time is essential

What do consumer device manufacturers want?



- Hardware
  - Minimize die size
  - Efficient memory bandwidth utilization
  - Minimize power; Maximize battery life

- Software
  - Choice of operating systems
  - Portability and reuse
  - Optimization at all levels
  - Validation

- $$
  - Least cost!

Software challenges facing consumer product design

Three major development pain points we hear from our customers in software enablement are:

1. Support of multiple hardware architectures
   - Keep costs low, Keep up with the latest technology, and Support more than one product line

2. Good tools story
   - Leverage software development tools across various products

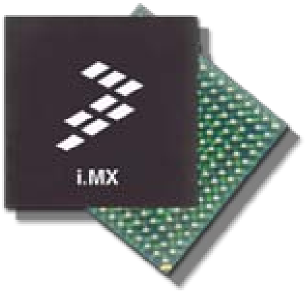
3. An overwhelming number of features to implement in a short development window

Contents

- ► Challenges facing consumer product design
- ► Embedded hardware and software platforms to address these challenges
- ► Software techniques to decrease boot time
- ► Overview of boot sequence
- ► Linux kernel optimizations

Hardware

- ► Each SoC needs to be defined for a target segment. Some of the customizations include:
  - Variations in Core speed
  - Multimedia and graphics capability
  - Cost of end device/platform
  - Automotive qualification
  - Industrial specifications



- ► Peripherals vary resulting in different board configurations
  - Device connectivity like Bluetooth, USB
  - Network connectivity like WiFi, Ethernet
  - Display variations – size and type
  - Storage variations – NOR, NAND, SD/MMC,..etc
  - Memory type – mDDR, DDR2, etc

i.MX Applications Processors

Multimedia:

Convergence of Audio, Video and Connectivity

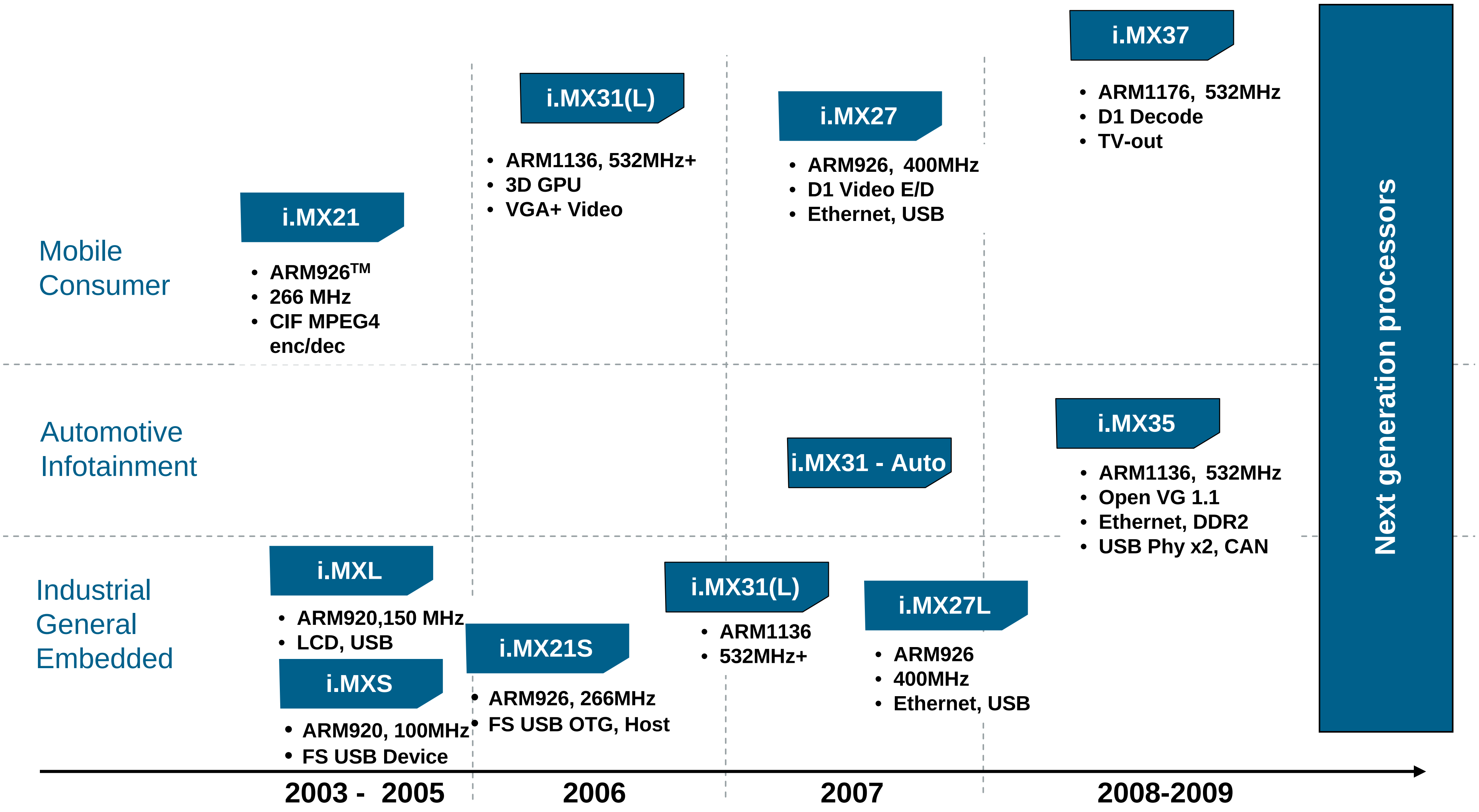
- **Primary Applications**
  - Media Players
  - Navigation Devices
  - Automotive Infotainment
  - General Embedded

- **Performance, Low Power and Portability**
  - Optimized performance per MHz
  - Low-power leadership
  - Range of audio and video formats, graphics and connectivity options
  - On-chip accelerators optimize performance and battery life
  - Linux Support

i.MX
Portable Consumer
Automotive
Industrial
Enterprise

Video* | *Graphics* | *Security* | *Audio* | *Connectivity* | *Low Power

Freescale Multimedia Applications Processors



Mobile Consumer

i.MX21
- ARM926[TM]
- 266 MHz
- CIF MPEG4 enc/dec

i.MX31(L)
- ARM1136, 532MHz+
- 3D GPU
- VGA+ Video

i.MX27
- ARM926, 400MHz
- D1 Video E/D
- Ethernet, USB

i.MX37
- ARM1176, 532MHz
- D1 Decode
- TV-out

Automotive Infotainment

i.MX31 - Auto

i.MX35
- ARM1136, 532MHz
- Open VG 1.1
- Ethernet, DDR2
- USB Phy x2, CAN

Industrial General Embedded

i.MXL
- ARM920,150 MHz
- LCD, USB

i.MXS
- ARM920, 100MHz
- FS USB Device

i.MX21S
- ARM926, 266MHz
- FS USB OTG, Host

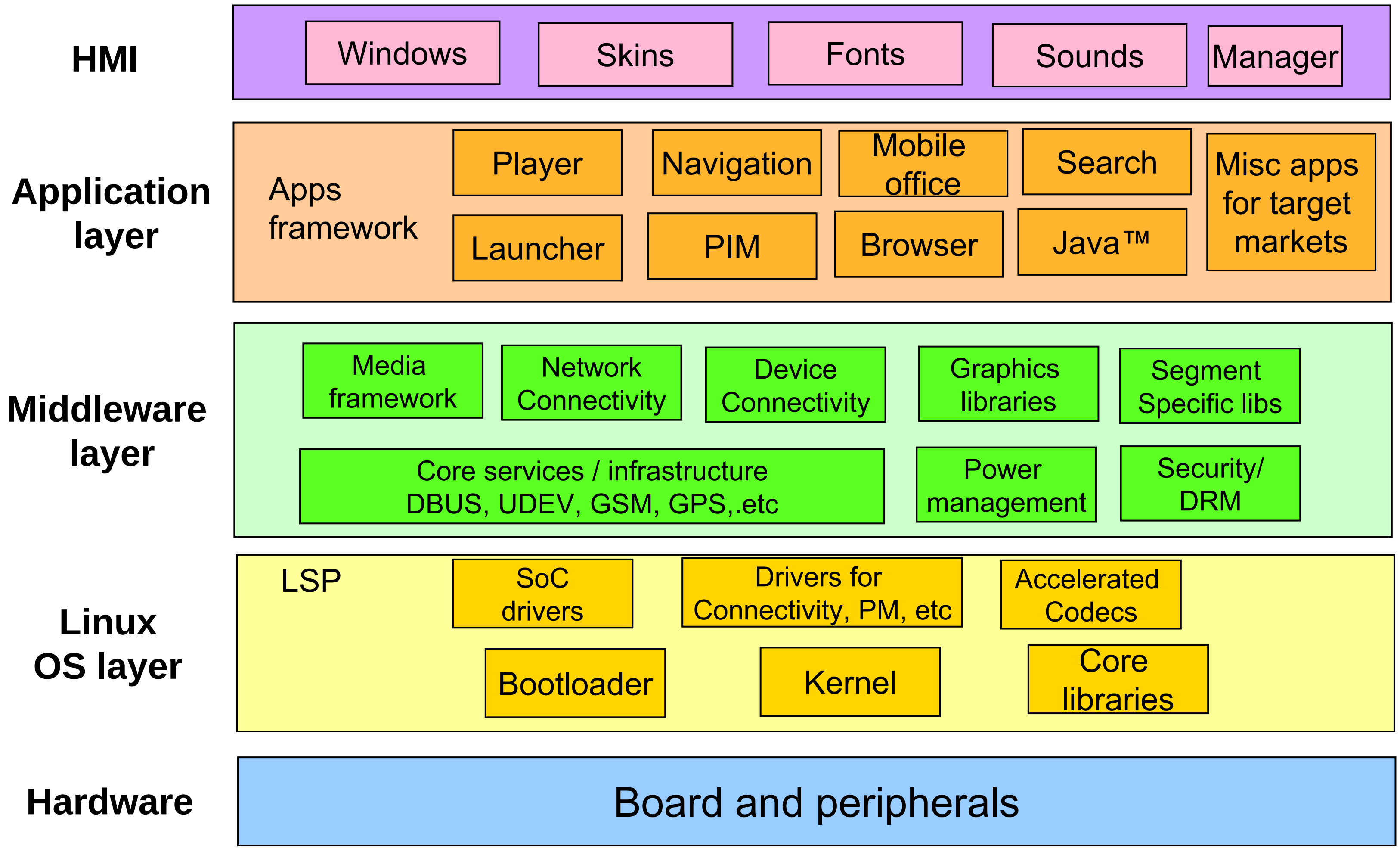
i.MX31(L)
- ARM1136
- 532MHz+

i.MX27L
- ARM926
- 400MHz
- Ethernet, USB

Next generation processors

2003 - 2005 | 2006 | 2007 | 2008-2009

The Linux software stack



HMI

Windows | Skins | Fonts | Sounds | Manager

Application layer

Apps framework

Player | Navigation | Mobile office | Search | Misc apps for target markets

Launcher | PIM | Browser | Java™

Middleware layer

Media framework | Network Connectivity | Device Connectivity | Graphics libraries | Segment Specific libs

Core services / infrastructure DBUS, UDEV, GSM, GPS,.etc | Power management | Security/ DRM

Linux OS layer

LSP

SoC drivers | Drivers for Connectivity, PM, etc | Accelerated Codecs

Bootloader | Kernel | Core libraries

Hardware

Board and peripherals

Contents

- ► Challenges facing consumer product design
- ► Embedded hardware and software platforms to address these challenges
- ► Software techniques to decrease boot time
- ► Overview of boot sequence
- ► Linux kernel optimizations

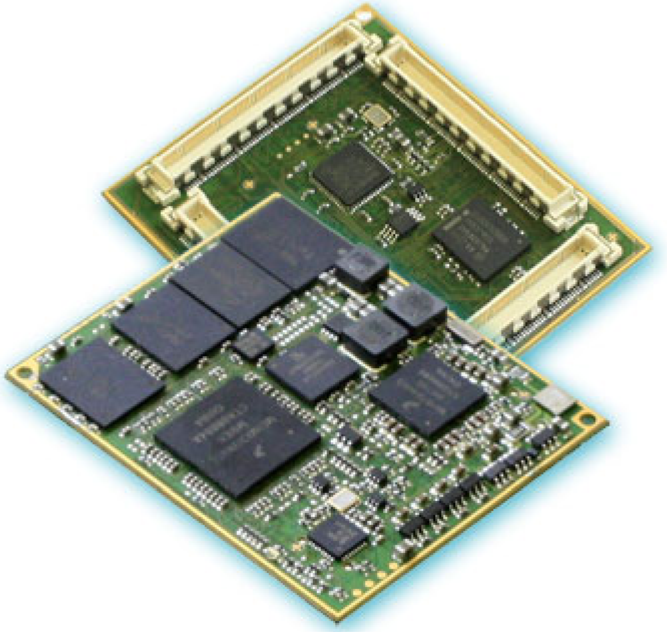
The Solution…

- ►Fast Boot is important to many markets
  - Consumer products
  - Automotive systems
  - Medical devices
- ►Boot time is affected by many factors:
  - Hardware design
  - Bootloader implementation
  - Kernel configuration
  - Application profile
- ►It is not difficult to get significant improvements with minimal investment

Hardware Design

Fast Boot Starts with Hardware Design

- Processor clock speed
- Clock generation
- DRAM interface
- Flash
- Power-on-reset circuitry
- Peripheral chips
- Configurable FPGAs, etc.
- and more…



Boatloader Considerations

- ►Bootloader implementation
- ►Bootloader has two primary responsibilities:
  - Initialize CPU/Hardware (minimally)
  - Locate, load and execute a kernel image
    - May involve several steps, including device i/o, decompression, etc.
- ►Most bootloaders have many more features
  - Not always a good thing...

Bootloader Consideration

- ►Bootloader Implementation
  - Lots of useful “development” functionality
    - dhcp, tftp, pci scan, mem utils
    - device initialization, Flash utilities, etc
    - In a production system, many of these features are unnecessary
  - Disabling these features can have a significant impact on boot time
  - For fastest boot, you want the bootloader to get out of the way as quickly as possible
  - Remember, small == fast

What components can we optimize?

►Bootloader

- Remove support for unused features
- Modify/remove hardware probing features
- Keep it Simple, Keep it Small

►Kernel

- Many opportunities for optimization
- Low hanging fruit can be easy to 'pluck'

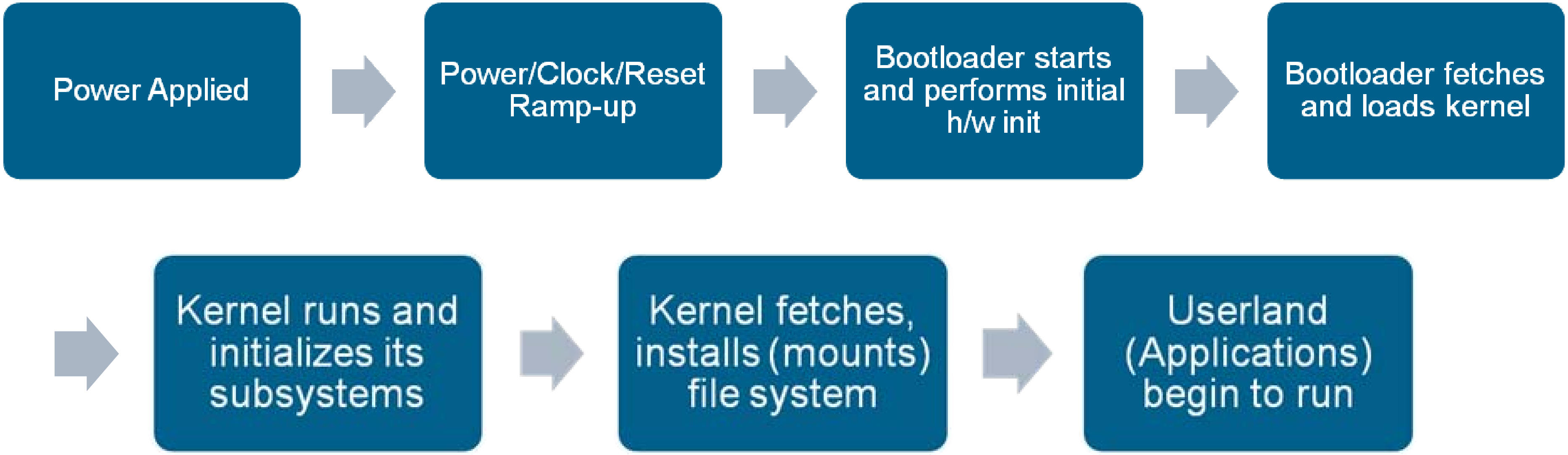
►Applications

- Most are up to you!

Contents

- Challenges facing consumer product design
- Embedded hardware and software platforms to address these challenges
- Software techniques to decrease boot time
- Overview of boot sequence
- Linux kernel optimizations

Typical Boot Sequence



Power Applied

Power/Clock/Reset Ramp-up

Bootloader starts and performs initial h/w init

Bootloader fetches and loads kernel

Kernel runs and initializes its subsystems

Kernel fetches, installs (mounts) file system

Userland (Applications) begin to run

Most of this is serial processing!

Much of userland early init is also serial processing!

Contents

- ► Challenges facing consumer product design
- ► Embedded hardware and software platforms to address these challenges
- ► Software techniques to decrease boot time
- ► Overview of boot sequence
- ► Linux kernel optimizations

Kernel Image Optimization

- ►Use Uncompressed Kernel
  - Decompression can take several seconds!
  - Tradeoff: more Flash storage required

- ►Kernel build produces two images*
  - Image and zImage (ARM)
  - Obviously, zImage is the compressed version
- ►On i.MX31, this saved on average ~750 ms

*Details vary for each architecture, ARM discussed here

Linux Kernel Configuration

- ►Eliminate Unnecessary Kernel Options
  - Reduces kernel size
  - Speeds up kernel loading

- ►Typical default kernel config contains lots of “stuff” you may not need:
  - MD/Raid support, IPv6, Numerous File Systems, Extended Partition support, etc.
  - Debug features such as kernel symbols, ikconfig, etc.
  - Many are compiled in features and increase kernel size

Examples: Interesting Kernel Configuration

- ►CONFIG_IKCONFIG
  - Removes support for config info, makes kernel smaller
  - (~ 250 ms improvement)
- ►CONFIG_MD
  - RAID/LVM support
- ►CONFIG_IDE
  - Saves init time if not used on HW w/ IDE ctrlr
  - Can also use hdx=noprobe

Examples: Interesting Kernel Configuration

- ► CONFIG_DEBUG_KERNEL
  - Reduces kernel size substantially
- ► CONFIG_KALLSYMS
  - Different than gcc -g
- ► CONFIG_PCCARD
  - Disable PCMCIA if not required
- ► Check Networking config options
  - Lots of functionality there, do you need it all?
    - ie. kernel autoconf, multicast, advanced router, tunnelling, etc.

More Interesting Kernel Config Options

- ► CONFIG_HOTPLUG
  - Remove support for hotplug if not required
- ► CONFIG_BUG
  - Used for debug – can be removed if desired
- ► Check Device Driver config options
  - Lots of default functionality that you may not need
  - Consumes space (and costs load time) even if not used
  - Can generate time-consuming h/w probes of non-existent devices
- ► Anything compiled as a module, if unused, is irrelevant
  - Won't affect start-up time
  - *Caveat: if you can avoid CONFIG_MODULES, kernel will be smaller, thus faster to load!*

More Interesting Kernel Config Options

- ►Remove support for unnecessary FS features
- ►Default configs often have much of this enabled (=y)
  - CONFIG_DNOTIFY
  - CONFIG_INOTIFY
  - CONFIG_XFS
  - CONFIG_AUTOFS4_FS (Automounter)
  - etc
- ►Won't make a large performance difference, but a smaller kernel will definitely load faster. (almost 20% smaller after removing unused FS features!)

XIP – Execute in Place

- Processor does not copy Kernel image to DRAM
  - Executes directly from (NOR) Flash
- Advantages
  - Reduces amount of DRAM required (and thus power)
  - Eliminates time-consuming copy from Flash
- Disadvantages
  - Depending on h/w architecture, could be much slower
    - i.e. burst/cache performance, etc.
  - Cost of Flash – kernel must be stored uncompressed
- Your Mileage May Vary

Calibration Routines

- Many hardware platforms spend considerable time in calibration routines
  - “Calculating BogoMips...”
  - Allows precise µdelay() routines
  - Can take significant time

- Use kernel command line: loops-per-jiffy:
  - lpj=xxxxx

- Easy to use: most platforms will display correct value in kernel log (and to console) on start-up

Driver Configuration

- ►Consider your system requirements:
  - What functionality must be available immediately?
  - What functionality can be deferred?

- ►Drivers can be pre-compiled into kernel or built as modules for loading later
  - Use pre-compiled drivers for those functions that must be immediately available
  - Use Loadable Modules for deferred functionality
    - Bear in mind the previous caveat: if you can deploy without loadable module support, smaller is faster!

File System Selection

- ►Consider CRAMFS for initial read-only File System
  - Compact and fast
  - No journaling entries to scan on initial mount
- ►Use tmpfs for /tmp, possibly /var, others
- ►Mount writable File System later, such as JFFS2 on NOR Flash
- ►Consider your tolerance to sudden power off
  - Journaling file systems can protect but at a cost of increased start-up times

Remove Support for printk()

- ►The “Brute Force” approach - CONFIG_PRINTK
  - Completely eliminates calls to printk()
- ►Advantages
  - Saves significant kernel size, and therefore load time
  - Eliminates many boot messages - decreasing boot time
- ►Disadvantage
  - No kernel status messages are available!
  - Makes kernel debugging very difficult
- ►A thoroughly tested kernel should work well here

Tools for Measuring Startup Time

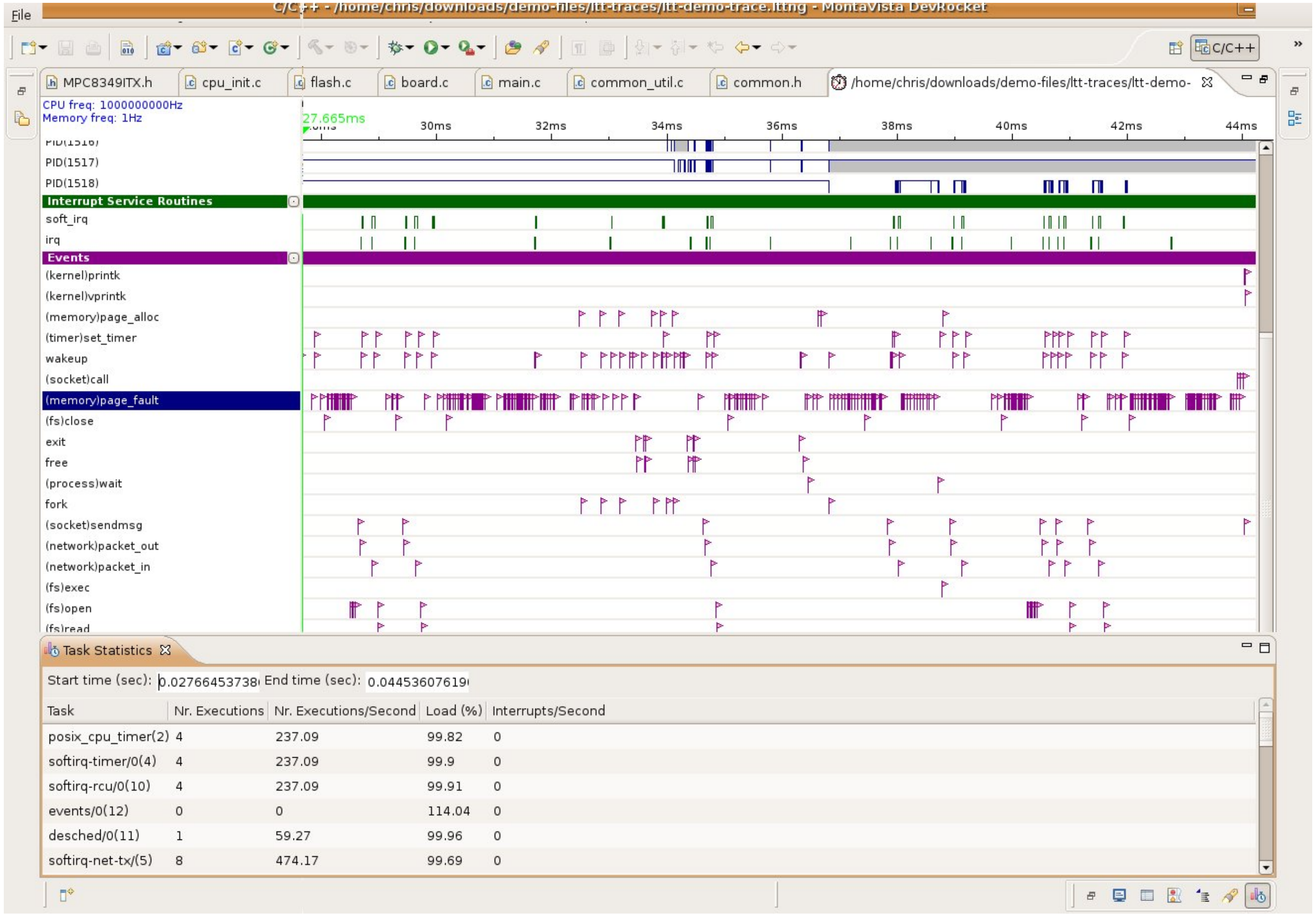
KFT: Kernel Function Timing

- Requires KALLSYMS mentioned above!
- Provides function call tracing and timing

```
 Entry      Duration   Local       Pid    Trace
---------- ---------- ---------- ------- --------------------------------
       162      11523          0       0 paging_init
       162      11523          0       0 |  free_area_init_nodes
       162      11523         12       0 |  |  free_area_init_node
       162      11511      11511       0 |  |  |  _etext+0x2f0
       162      11511          0       0 __alloc_bootmem_node
       162      11511      11511       0 !  __alloc_bootmem_core
     11787       2307       2307       0 vfs_caches_init_early
     11787       1531         69       0 vfs_caches_init_early
     11787        686          0       0 [  alloc_large_system_hash
     11787        686          0       0 [  [  __alloc_bootmem
     11787        686          0       0 [  [  [  __alloc_bootmem_nopanic
     11787        686        686       0 [  [  [  [  __alloc_bootmem_core
     13318        776        776       0 [  inode_init_early
     14094       1208        641       0 mem_init
     14094        567          4       0 #  free_all_bootmem
     14094        563        563       0 #  #  free_all_bootmem_core
     15607       3851       3851       0 schedule
     15607       3848       3848       0 schedule
     15630    1573099    1573099       1 kernel_init
     15666      81152      81152       4 schedule
     15666      81139      81139       4 schedule
```

Tools for Measuring Startup Time

Linux Trace Toolkit



Start time (sec): 0.02766453738 End time (sec): 0.04453607619

Task	Nr. Executions	Nr. Executions/Second	Load (%)	Interrupts/Second
posix_cpu_timer(2)	4	237.09	99.82	0
softirq-timer/0(4)	4	237.09	99.9	0
softirq-rcu/0(10)	4	237.09	99.91	0
events/0(12)	0	0	114.04	0
desched/0(11)	1	59.27	99.96	0
softirq-net-tx/(5)	8	474.17	99.69	0

Initial Baseline

- ►i.MX31 Baseline
- ►Initial Software Configuration
  - Redboot Version FSL 200740
  - Linux 2.6.21 – default config
  - Typical embedded configuration
    - ▪ Networking, DHCP client
    - ▪ Ext2, JFFS2 File System support
    - ▪ Full SysV init
- ►Time to Boot:
  - 0:36 seconds, kernel start to command prompt
  - Redboot bootloader takes 10 seconds (not bad for stock hardware platform!)

After Inexpensive Optimizations

►Final Configuration

- Networking, static IP
- Busybox userland
- Many kernel optimizations

►Total boot time

- 2.7 seconds, kernel start to command prompt!
- Remember, this improvement came at a very modest engineering cost (i.e. effort!)

Other Useful Ideas

- linux-embedded (mail list)
  - Proposal for deferred initcalls
- /proc/uptime
  - System uptime/idle time. Useful for scripting
- printk time stamps (see next slide)
- initcall_debug (example to follow)
- quiet
  - on kernel command line, suppresses printk output during boot, preserving the printk infrastructure

Courtesy of Tim Bird
Architecture Group Chair, CE Linux Forum
Senior Staff Engineer, Sony Corporation of America

Other Useful Ideas

- ►printk timestamps (CONFIG_PRINTK_TIME)
  - Appends time info to printk() output
  - Enables measurement of long operations, esp. at boot time

```
[ 1.321054] md: linear personality registered for level -1
[ 1.326629] md: raid0 personality registered for level 0
[ 1.331964] md: raid1 personality registered for level 1
[ 1.342289] TCP cubic registered
[ 1.345936] NET: Registered protocol family 1
[ 1.350403] NET: Registered protocol family 17
[ 1.355816] RPC: Registered udp transport module.
[ 1.360571] RPC: Registered tcp transport module.
[ 1.366034] drivers/rtc/hctosys.c: unable to open rtc device (rtc0)
[ 2.880506] IP-Config: Complete:
[ 2.883575]  device=eth0, addr=192.168.1.201, mask=255.255.255.0, gw=255.255.255.255,
[ 2.892227]  host=8349itx, domain=, nis-domain=(none),
[ 2.897798]  bootserver=192.168.1.9, rootserver=192.168.1.9, rootpath=
[ 2.906152] md: Autodetecting RAID arrays.
```

Other Useful Ideas

▶initcall_debug

- Great way to get a detailed view of system init timing
- Simply add “initcall_debug” to kernel command line

```
4 msecs: initcall c02c7da0 t linear_init
5 msecs: initcall c02c6bbc t init_sd
7 msecs: initcall c02cc450 t init_sunrpc
10 msecs: initcall c02c01a8 t slab_sysfs_init
15 msecs: initcall c02c8d50 t genl_init
24 msecs: initcall c02c55c4 t serial8250_init
30 msecs: initcall c02c6364 t gfar_init
34 msecs: initcall c02c743c t physmap_init
72 msecs: initcall c02c9c60 t inet_init
127 msecs: initcall c02c4e4c t pty_init
4597 msecs: initcall c02cabe0 t ip_auto_config
```

Additional Ideas

- Parallelize init tasks using custom startup scripts
- Provide user feedback early (i.e flash screens, etc) to give the impression that the unit is booted while other work is being done in the background.
- Use a pre-configured hibernate image

Promotional Discounts

25% off Freescale
i.MX31 PDK development platform

20% off MontaVista Mobilinux
5.0 LSP for i.MX31

Offers good until Feb 18th.
Must be *new* customer with valid business email address – otherwise email sales@mvista.com.
Please fill out survey information and promo information will be emailed to you.
Does not apply to attendees of recorded web seminar.

Thank You!

Event archive available at:

http://ecast.opensystemsmedia.com/

E-mail us at: ecast@opensystemsmedia.com