Optimizing Linux System

Chase Maupin

ASP Catalog Applications

Derived from presentation by Juan Gonzales and Brijesh Singh

Texas Instruments

Objective

- Speed up your software
  - Reduce the system boot time
- Save money
  - Reduce the amount of memory (DDR and Flash) required
- How? **Optimizing the software**

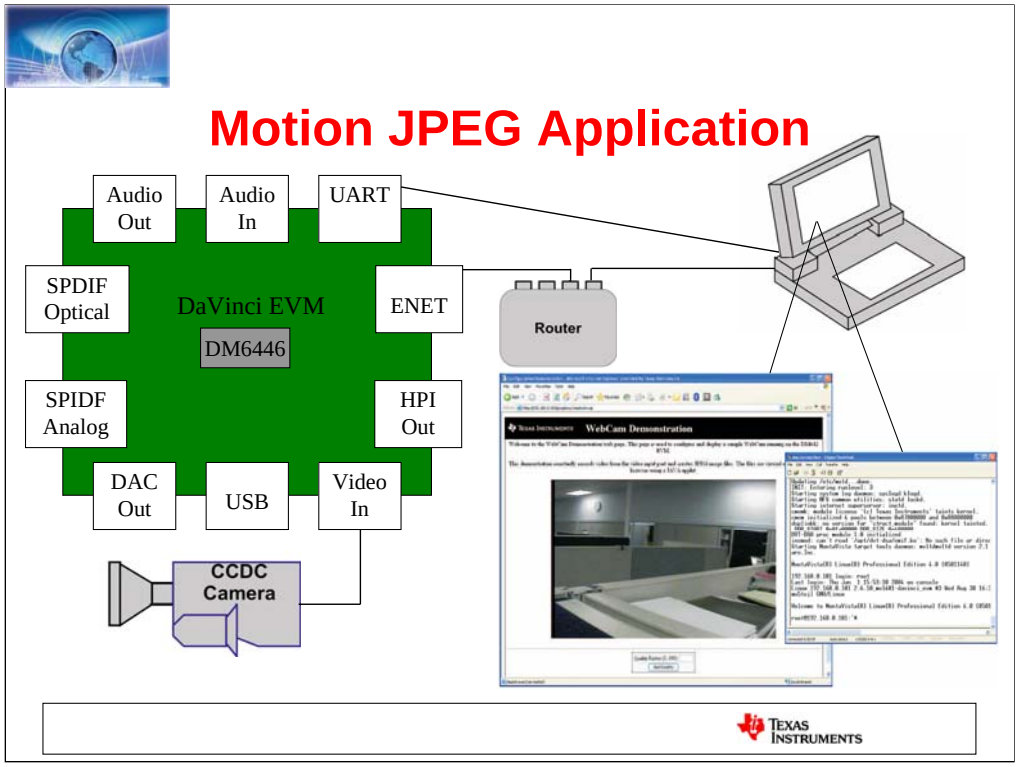
Agenda

- **Overview of Motion JPEG application**
- Optimizing Linux Kernel and File System
- Configuring DSP algorithm
- Optimized Motion JPEG Server demo

3

Texas Instruments

-emphasize demo will be shown at the end



-If you cannot read the peripheral text, don't have it there

Motion JPEG Application

- ***Motion JPEG Demo on TMS320DM6446*** (**spraah9a**)
- Customer end-goal: Optimize all areas of application
  - Linux Kernel
  - File System
  - DSP Algorithm

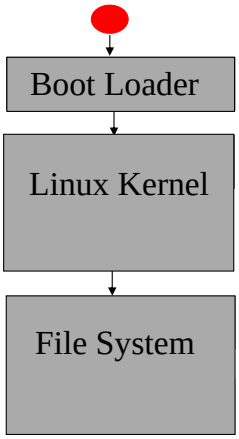
Agenda

- Overview of Motion JPEG application
- **Optimizing Linux Kernel and File System**
- Configuring DSP algorithm
- Optimized Motion JPEG Server demo

Embedded Linux Optimization

- Linux Boot Process Overview
- Optimizing Linux Kernel
  - For Size
  - For Speed
- Optimizing File System
  - For Size
  - For Speed
- DevRocket

Linux Boot Process



Power On

Bootloader Code

Initialize hardware

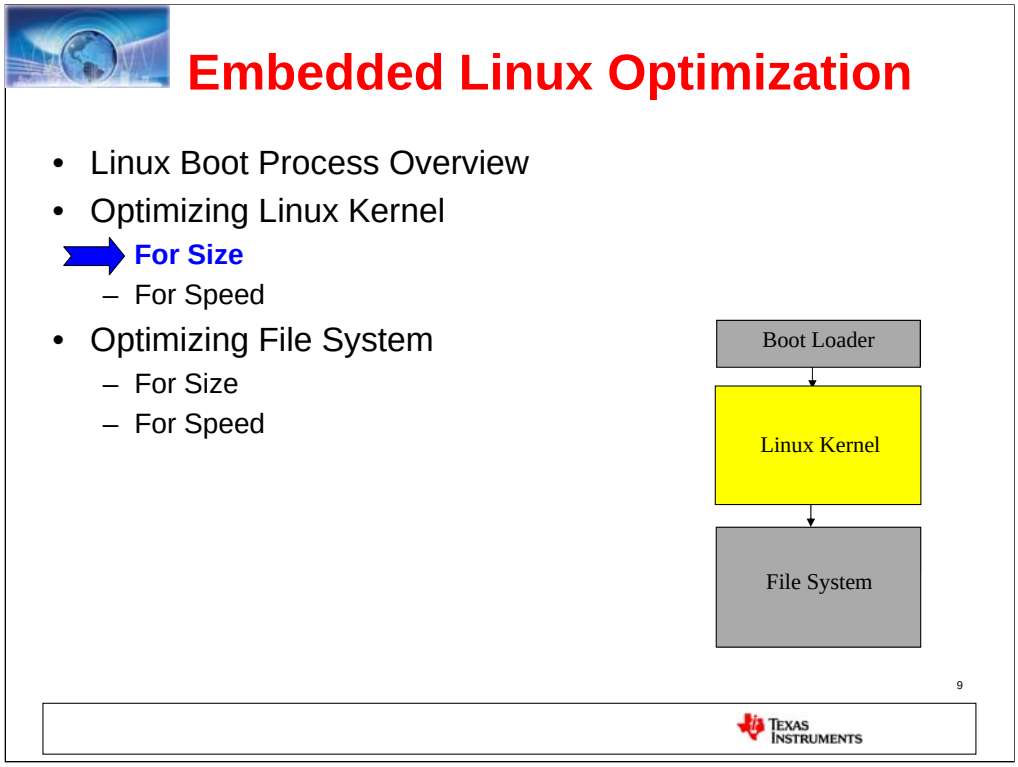
Initial RAM Disk

/sbin/init – 1st process exe by kernel

Login console

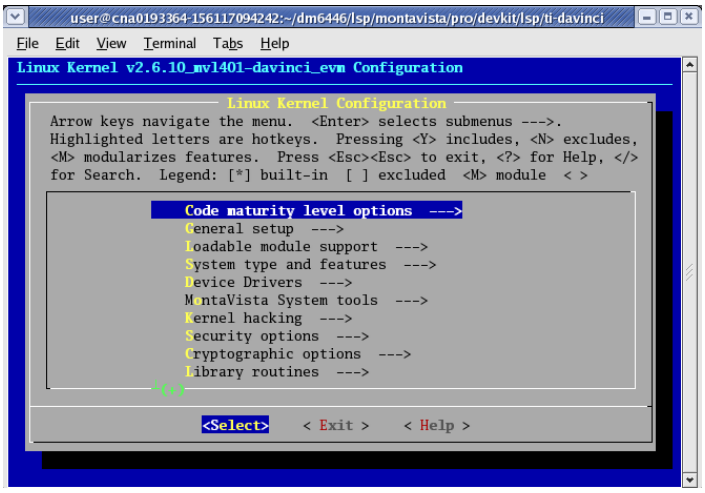
8

Texas Instruments



Needs work (High-light)

Linux Kernel Configuration



> make ARCH=arm CROSS_COMPILE=arm_v5t_le- menuconfig

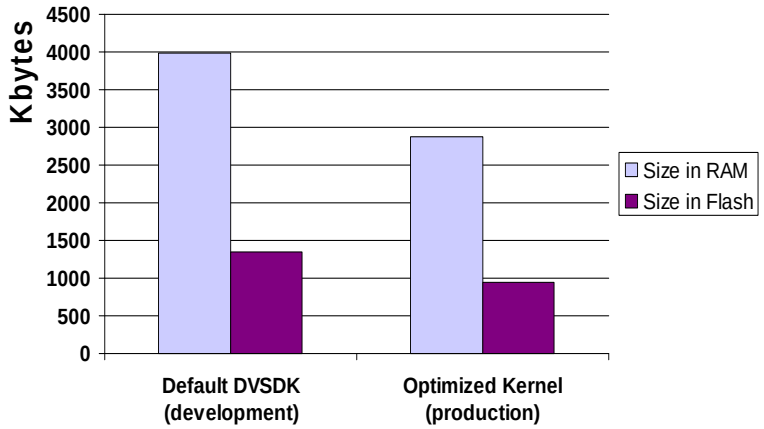
Remove unneeded features

Goal: Smallest kernel necessary to run JPEG Server demo

Remove following important features

- ATA support
- USB support
- SCSI support
- Input device support
- Ext3 file system support
- OSS
- Frame buffer support

Reduced demo kernel size

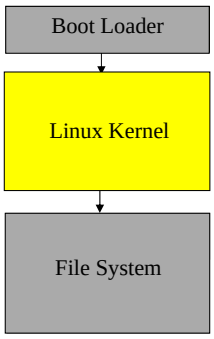


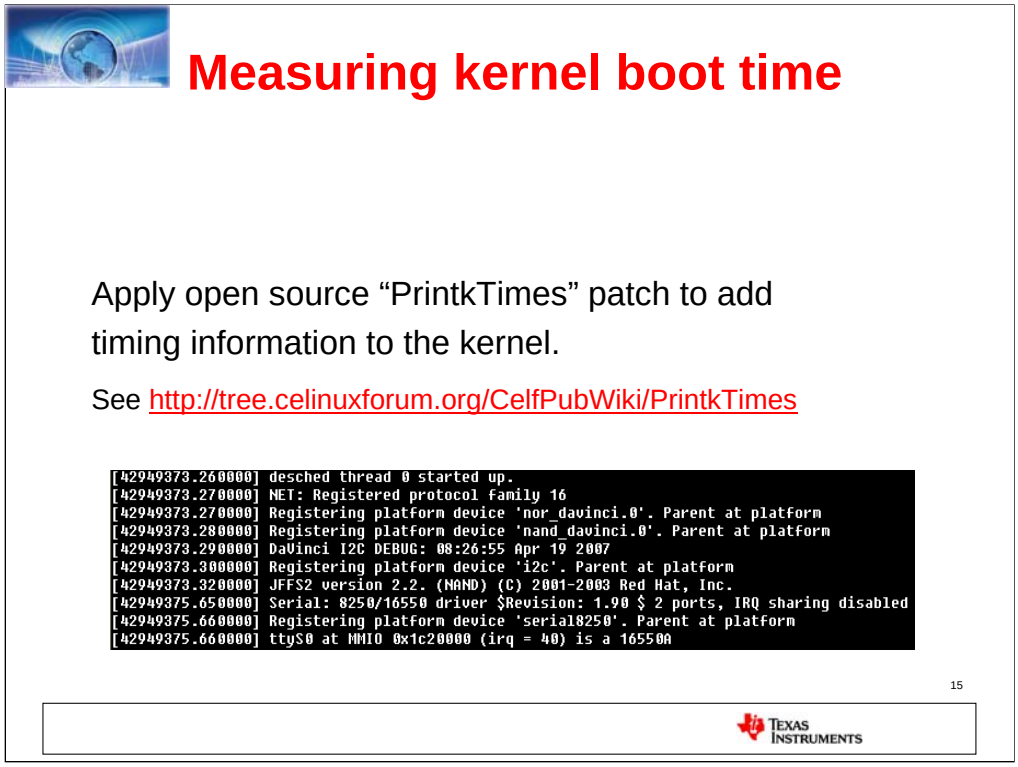
More Ideas

- Remove kernel messages (printk,…)
- TinyLinux patches to remove kernel features
  - http://www.selenic.com/linux-tiny
- Boot without sysfs support
  - http://www.selenic.com/linux-tiny/index.cgi/BootingWithoutSysFs
- Kernel XIP (eXecute In Place)
  - http://elinux.org/Kernel_XIP
- Eliminate excess inlining
- Initramfs instead of initrd
  - No artificial size limits or caching
  - See Documentation/filesystems/ramfs-rootfs-initramfs.txt in kernel tree

Embedded Linux Optimization

- Linux Boot Process Overview
- Optimizing Linux Kernel
  - For Size
  - **For Speed**
- Optimizing File System
  - For Size
  - For Speed

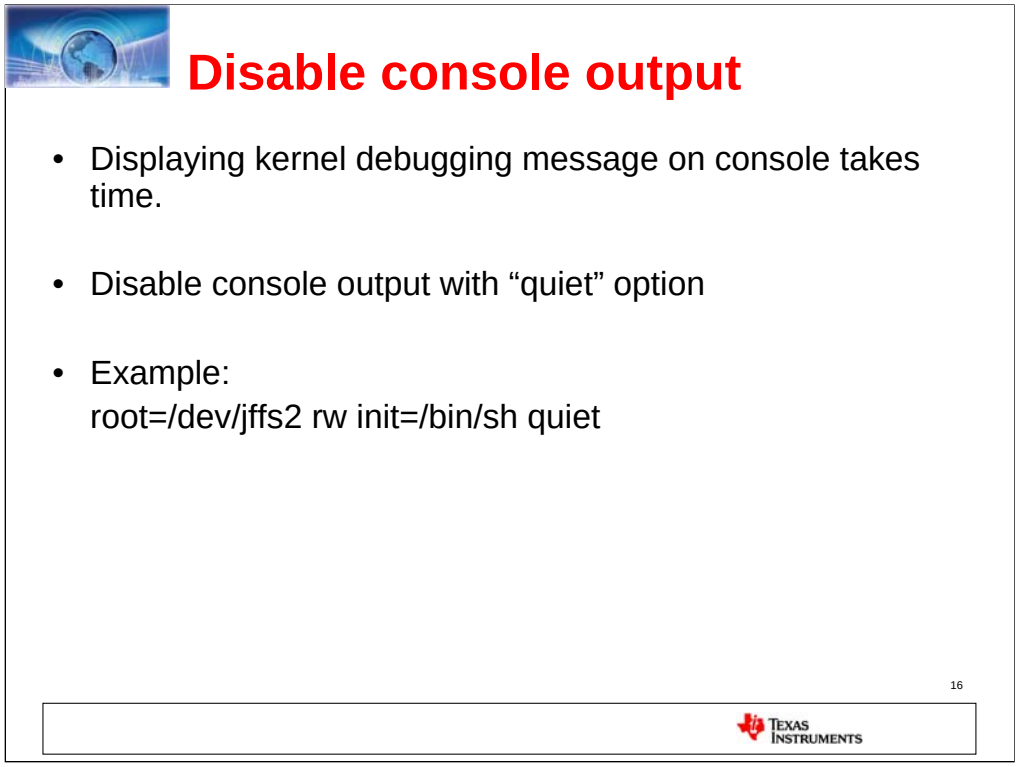




"Printk-times" is a simple technology which adds some code to the standard kernel printk routine, to output timing data with each message. While crude, this can be used to get an overview of the areas of kernel initialization which take a relatively long time. This feature is used by the Bootup Time Working Group to identify areas of the Linux kernel requiring work to improve bootup time, and to measure the improvements of changes made by the working group. The technology for this feature consists of a patch and a utility program. The patch alters the printk code in the kernel to emit the timing data.

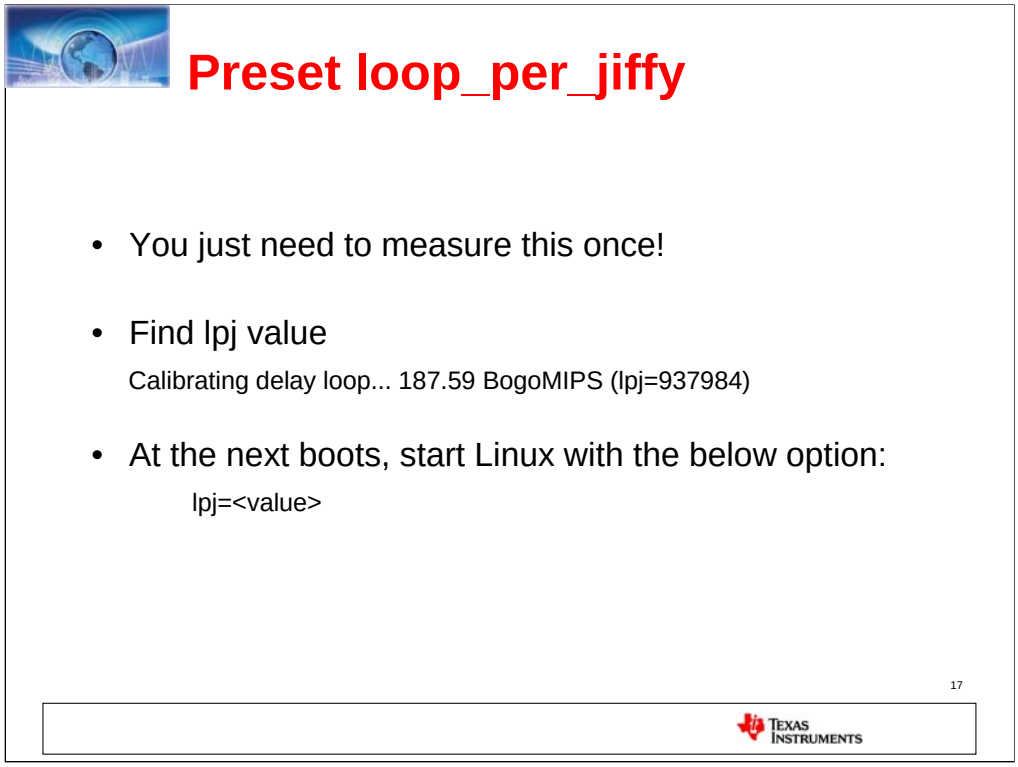
UPDATE: The patch was incorporated into the mainline kernel as of version 2.6.11! Both the feature, and the utility program are now part of mainline Linux!!

With printk-times turned on, the system emits the timing data as a floating point number of seconds (to microsecond resolution) for the time at which the printk started. The utility program shows the time between calls, or it can show the times relative to a specific message. This makes it easier to see the timing for specific segments of kernel code.



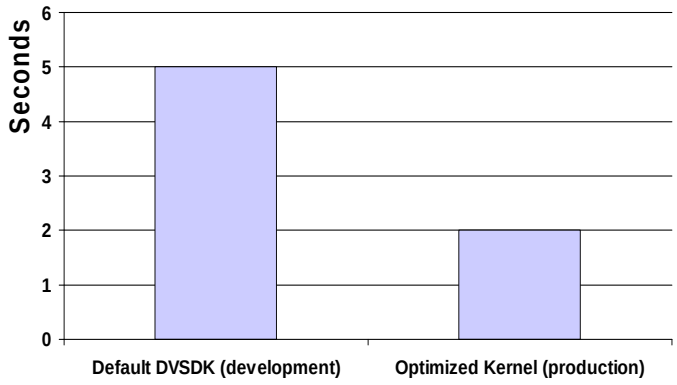
You can save time during kernel bootup by disabling the console output. The easiest way to do this is to use the "quiet" option on the kernel command line (described below). Printk output is usually directed to a serial port or a VGA console during bootup. By disabling console output, the time taken to output the characters (and perform things like software scrolling of the display buffer) is eliminated.

This saves time during kernel bootup by suppressing printk output. Printk output delays depend on a number of factors, but in the use cases cited below, the savings were in the range of a few hundred milliseconds. With a serial console, the time to output characters is dependent on the serial port speed. However, with a VGA console, the time to output the characters is dependent on the speed of the processor. Therefore, the slower your processor, the more savings you will gain from this technique.



At each boot, the Linux kernel calibrates a delay loop (for the udelay function). This measures a loops_per_jiffy (lpj) value. This takes about 25 jiffies (1 jiffy = time between 2 timer interrupts).In embedded systems, it can be about 250 ms!

Reduced kernel boot time



More Ideas

- Copy kernel from flash to RAM using DMA
- Reduce probe time on some IDE
- Kernel XIP (eXecute In Place)
- Use as many module as possible

19

Texas Instruments

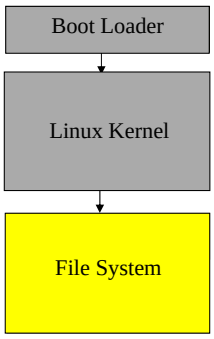
-It's necessary to copy binary images, such as a kernel image, file system images, and so on, from ROM to RAM on bootup if XIP isn't used. In this case, using DMA transfer is very efficient to save the time and CPU resources.

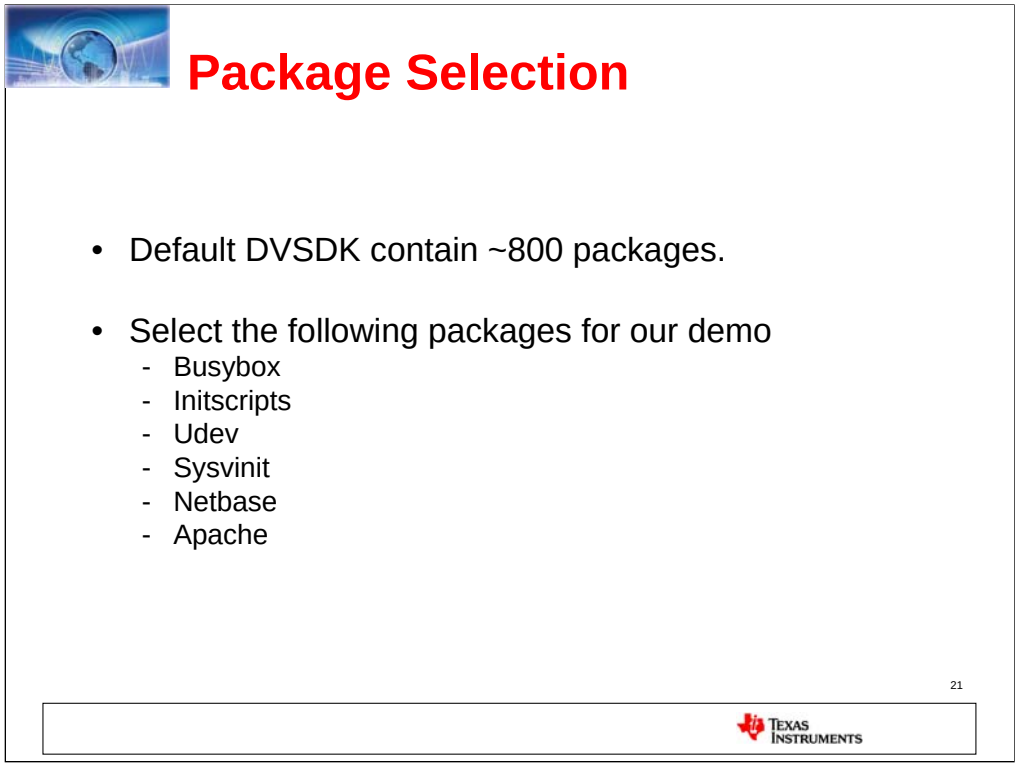
-I t was noted on a test machine that IDE initialization takes a significant percentage of the total bootup time. Almost all of this time is spent busywaiting in the routine ide_delay_50ms(). It is trivial to modify the value of the delay used in this routine. Reducing the duration of the delay in the ide_delay_50ms() routine provides a substantial reduction in the overall bootup time for the kernel on a typical desktop system. It also has potential for use in embedded systems where PCI-based IDE drives are used.

- When the kernel is executed in place, the bootloader does not have to 1) read the kernel from flash, 2) decompress the kernel, and 3) write the kernel to RAM.

Embedded Linux Optimization

- Linux Boot Process Overview
- Optimizing Linux Kernel
  - For Size
  - For Speed
- Optimizing File System
  - **For Size**
  - For Speed





•Busybox - It provides minimalist replacements for the most common utilities you would usually find on your desktop system (i.e., ls, cp, mv, mount, tar, etc.). The utilities in BusyBox generally have fewer options than their full-featured GNU cousins; however, the options that are included provide the expected functionality and behave very much like their GNU counterparts.

• initscript - The initscripts package contains the basic system scripts used to boot your MV Linux system, change run levels, and shut the system down cleanly. Initscripts also contains the scripts that activate and deactivate most network interfaces.

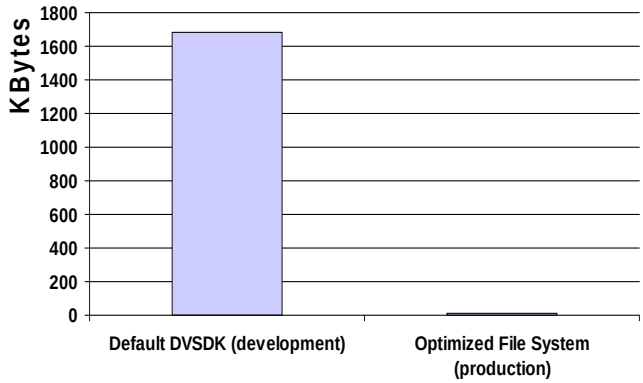
•Udev deamon for user space

• sysvinit- The SysVinit package contains a group of processes that control the very basic functions of your system. SysVinit includes the init program, the first program started by the Linux kernel when the system boots. Init then controls the startup, running and shutdown of all other programs.

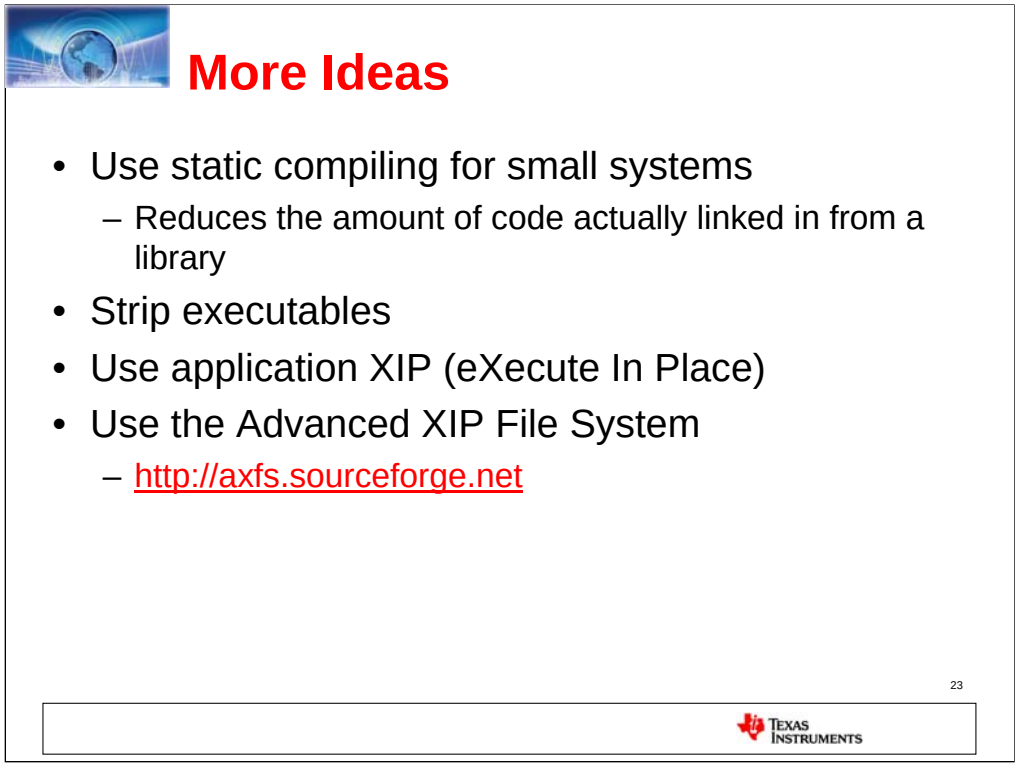
•Netbase – basic networking scripts

•apache - Http web server

Reduced demo file system size



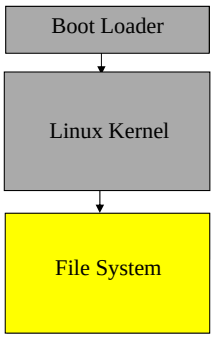
Texas Instruments



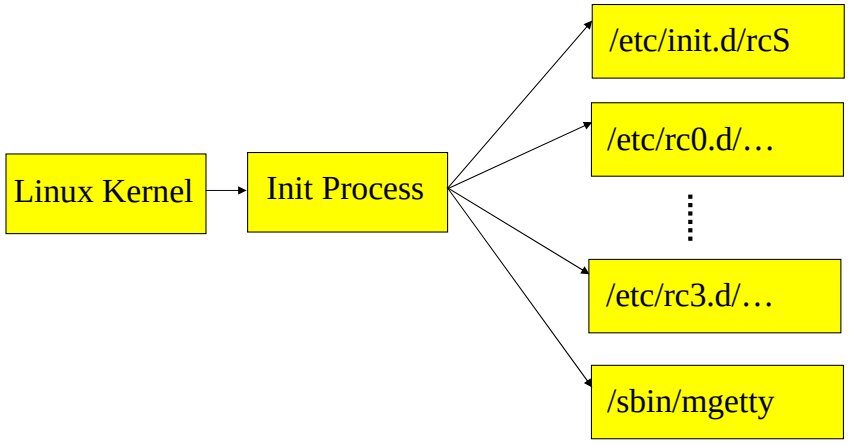
Strip - Compiled executables and libraries contain extra information which can be used to investigate problems in a debugger. This was useful for the tool developer, but not for the final user. To remove debugging information, use the strip command. This can save a very significant amount of space!

Embedded Linux Optimization

- Linux Boot Process Overview
- Optimizing Linux Kernel
  - For Size
  - For Speed
- Optimizing File System
  - For Size
  - **For Speed**



Starting System Services



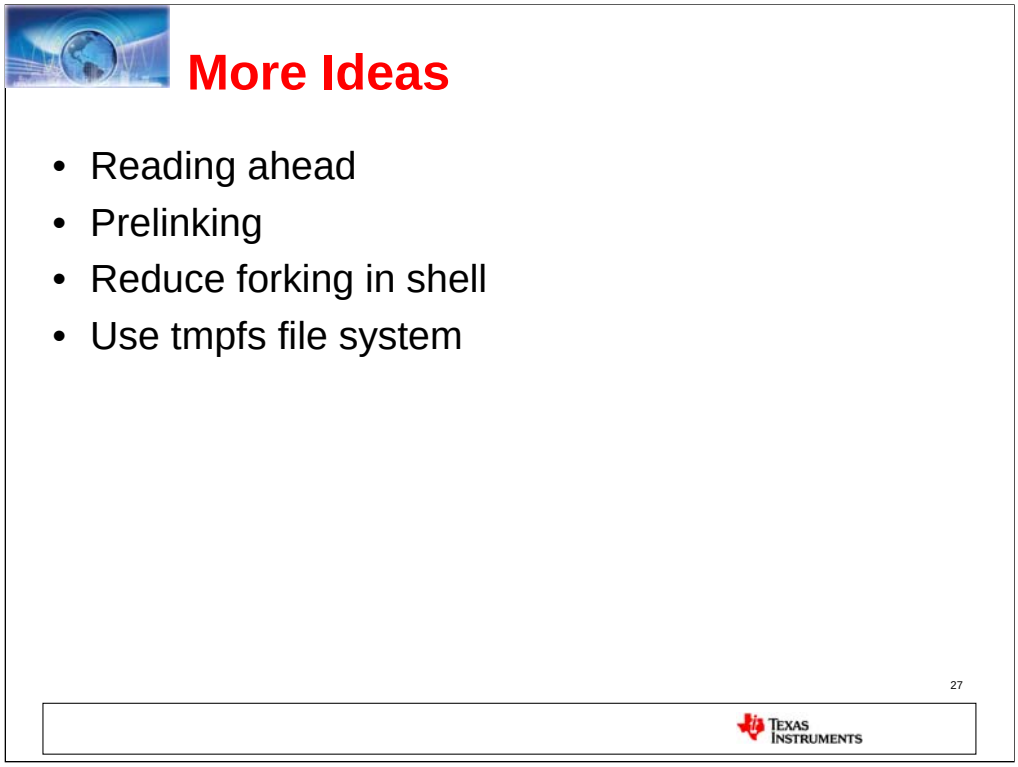
Starting System Services

- SysVinit

  Starts services sequentially.Waits for the current startup script to be complete to start the next one!
- Initng http://initng.thinktux.net

  Start services in parallel, as soon as their preconditions are met.

You can hunt system startup trouble by using bootchart program (www.bootchart.org)



Reading Ahead - Linux keeps the contents of all the files it reads in RAM (in the *page cache*), as long as it doesn't need the RAM pages for something else. Idea: load files (programs and libraries in particular) in RAM cache

before using them. Best done when the system is not doing any I/O. Thanks to this, programs are not stuck waiting for I/O. Used the Knoppix distribution to achieve very nice boot speedups. Also planned to be used by Initng. Not very useful for systems with very little RAM: cached pages are recycled before the files are accessed.

Prelinking - To load and start an executable, the dynamic linker has a significant amount of work to do (mainly address relocation) It can take a lot of time for executables using many shared libraries! In many systems in which executables and shared libraries never change, the same job is done every time the executable is started.

• prelink modifies executables and shared libraries to simplify the dynamic linker relocation work.

•This can greatly reduce startup time for big applications (50% less for KDE!). This also saves memory consumed by relocations.

•Can be used to reduce the startup time of a Linux system.

•Just needs to be run again when libraries or executables are updated

Agenda

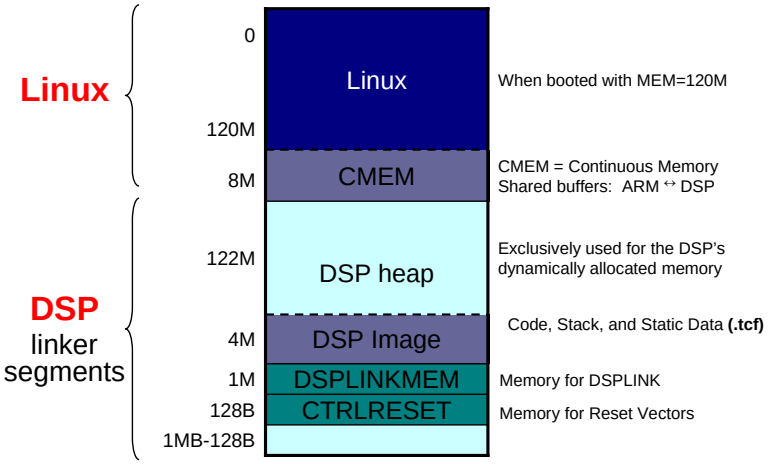
- Overview of Motion JPEG application
- Optimizing Linux Kernel and File System
- **Configuring DSP algorithm**
- Optimized Motion JPEG Server demo

Configuring DSP algorithm

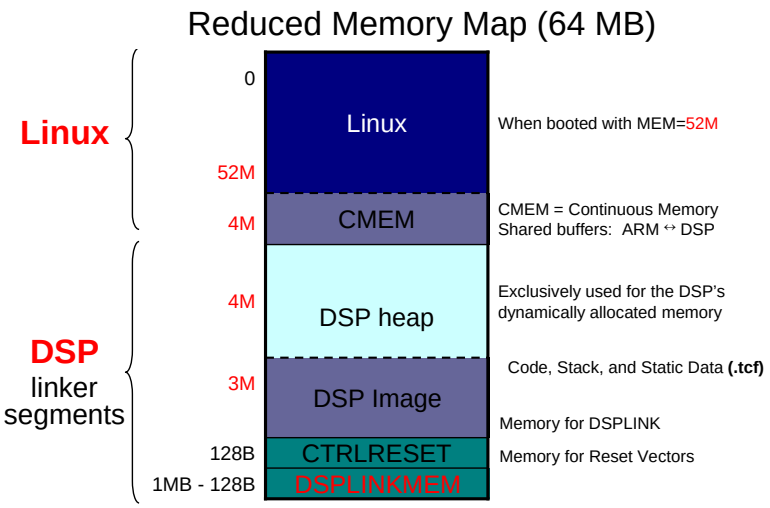
- Goal: DSP Image (algorithm) that runs on smaller memory footprint (e.g. 64MB Vs 256MB by default)
- Overview of Process http://wiki.davincidsp.com
- TI and ASPs are here to help!

Optimizing DSP Server

DVEVM Default DDR2 Memory Map (256MB)



Optimizing DSP Server



DSP Server Optimization Links

- http://wiki.davincidsp.com/index.php?title=Changing_the_DVEVM_memory_map
- http://wiki.davincidsp.com/index.php?title=Codec_Engine_configuration_en_breve

Agenda

- Overview of Motion JPEG application
- Optimizing Linux Kernel and File System
- Configuring DSP algorithm
- **Optimized Motion JPEG Server demo**

-

Optimized JPEG Server demo

Resources

- Inside Linux Boot Process
  http://www.ibm.com/developerworks/linux/library/l-linuxboot/index.html
- Boot Time Resource
  http://tree.celinuxforum.org/pubwiki/moin.cgi/BootupTimeResources
- Embedded Linux Resources
  http://www.elinux.org/
- Davinci wiki
  http://www.wiki.davincidsp.com