Presented by : Grégory Clément – Linux / Open Source manager – gclement@adeneo.eu

Embedded Linux Conference Europe 2009

Embedded Experts dedicated to the success of your design

How we got a 3D application booting in 5 seconds under Linux

Adetel Group

Adeneo Embedded

Agenda

Embedded Experts dedicated to the success of your design

- The Problem
- The context
- The boot process
- Time measurement
- Techniques for optimizations
- Results

Introduction

Embedded Experts dedicated to the success of your design

- About Adeneo
  - French group
  - Foreign subsiadiaries
    - USA, China, Africa
  - Developping and manufacturing electronics products
  - Developping embedded softwares
    - Application software
    - Drivers development, BSP (Linux, CE)
  - Works with embedded Linux for + 7 years
    - Involved in the community (ex : Xenomai, RTAI)
    - Interested in improving bootup time in our systems

Introduction

Embedded Experts dedicated to the success of your design

- About us
  - Grégory Clément
  - Simon Polette

The problem

Embedded Experts dedicated to the success of your design

- People don't want to wait
- Especially with embedded devices

Hardware

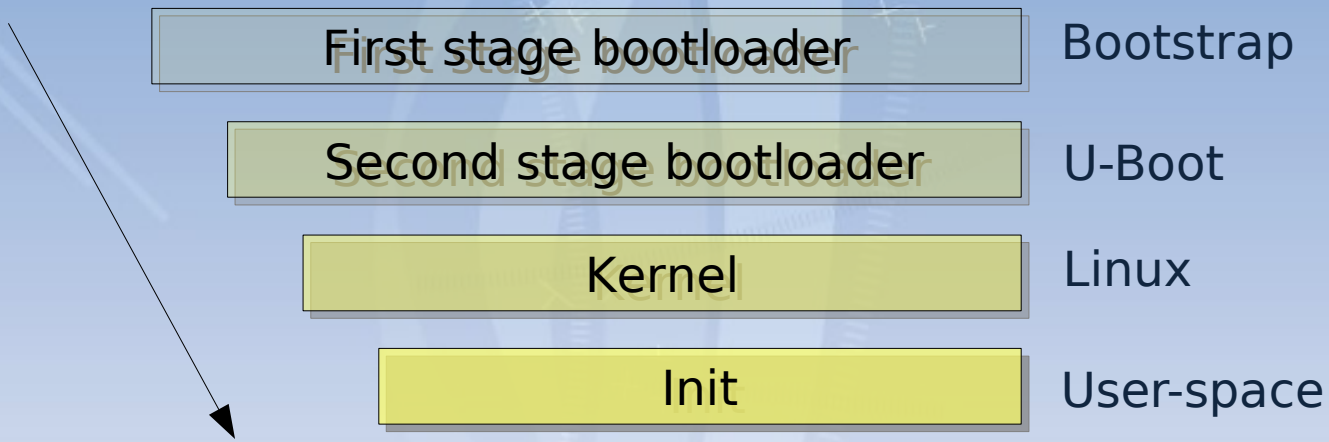
Embedded Experts dedicated to the success of your design

- AT92SAM9261-EK board :
  - ARM9-based µC at 200MHz
  - 64MB SDRAM
  - 256MB NAND Flash
  - DataFlash (SPI)

The boot process

The boot process

Embedded Experts dedicated to the success of your design



The boot process

Embedded Experts dedicated to the success of your design

- Bootstrap :
  - Basic hardware init. (GPIO, Clock, SDRAM, etc)
  - Load and start U-Boot
- U-Boot :
  - Hardware initializations
  - Load and start kernel
- Kernel :
  - Drivers initializations
  - Mount root filesystem
- Init :
  - First User-space processus
  - Launch RC scripts
  - Start graphic environment

Time measurement

Time measurement

Embedded Experts dedicated to the success of your design

- Find where the system loose time
- Be able to trace the boot process
- Identify the worst problem first

Adetel Group

Adeneo Embedded

Time measurement

Embedded Experts dedicated to the success of your design

- Global measurement
  - serialchrono
- Kernel measurement
  - printk-times
  - initcall_debug
- User-space measurement
  - Bootchart-lite

Time measurement

Embedded Experts dedicated to the success of your design

Global measurement

Adetel Group

Adeneo Embedded

Serialchrono

Embedded Experts dedicated to the success of your design

- Simple script
- Capture serial output, add time stamp at begining of each line, then echo it.
- No need for any other module to run it, just /bin/sh
- Usage :

```
./serialchrono TTYS="/dev/ttyS0"
```

Serialchrono

Embedded Experts dedicated to the success of your design

- Serialchrono output example :

```
[    1.252419] Area 0:  C0000000 to C00041FF (RO) Bootstrap
[    1.266907] Area 1:  C0004200 to C00083FF      Environment
[    1.274590] Area 2:  C0008400 to C0041FFF (RO) U-Boot
[    1.281892] Area 3:  C0042000 to C0251FFF      Kernel
[    1.289026] Area 4:  C0252000 to C083FFFF      FS
[    1.296206] In:    serial
[    1.303194] Out:   serial
[    1.310185] Err:   serial
[    1.317642] Hit any key to stop autoboot:  0
[    1.984084] ## Booting kernel from Legacy Image at 22000000 ...
[    1.992148]    Image Name:   Linux-2.6.31-rc2
[    1.999412]    Image Type:   ARM Linux Kernel Image (uncompressed)
[    2.007201]    Data Size:    911700 Bytes = 890.3 kB
[    2.013851]    Load Address: 20008000
[    2.021067]    Entry Point:  20008000
[    2.148271]    Verifying Checksum ... OK
[    2.268042]    Loading Kernel Image ... OK
[    2.275552] OK
[    2.282747]
[    2.289796] Starting kernel ...
```

Adetel Group

Adeneo Embedded

Serialchrono

Embedded Experts dedicated to the success of your design

- Advantages :
  - Very simple to use, no need installation
  - "Passive" instrumentation (just read what the target send)
  - Doesn't add any overhead
- Disadvantages :
  - First printks are queued up and printed only once serial console have been initialized
  - Timestamp only what is printed in console

Time measurement

Embedded Experts dedicated to the success of your design

Kernel measurement

Adetel Group

Adeneo Embedded

Kernel measurement

Embedded Experts dedicated to the success of your design

- Printk-time
  - Add timestamp at the begining of each console output line
  - To activate it, just select the option in kernel configuration, or just add `printk.time=1` in the kernel command line

```
# Kernel hacking
#
CONFIG_PRINTK_TIME=y
```

Kernel measurement

Embedded Experts dedicated to the success of your design

- Printk-time output example :

```
[    0.000000] NR_IRQS:192
[    0.000000] AT91: 96 gpio irqs in 3 banks
[    0.000000] Console: colour dummy device 80x30
[    0.000000] console [ttyS0] enabled
[    0.120000] Calibrating delay loop (skipped) preset value.. 124.51 BogoMIPS (lpj=622592)
[    0.130000] Mount-cache hash table entries: 512
[    0.130000] CPU: Testing write buffer coherency: ok
[    0.140000] NET: Registered protocol family 16
[    0.180000] bio: create slab <bio-0> at 0
[    0.190000] SCSI subsystem initialized
[    0.200000] usbcore: registered new interface driver usbfs
[    0.200000] usbcore: registered new interface driver hub
[    0.210000] usbcore: registered new device driver usb
[    0.230000] NET: Registered protocol family 2
[    0.230000] IP route cache hash table entries: 1024 (order: 0, 4096 bytes)
[    0.240000] TCP established hash table entries: 2048 (order: 2, 16384 bytes)
```

- Note :
  - Timestamps are available only once clock is initialized

initcall_debug

Embedded Experts dedicated to the success of your design

- Measure time spend in each 'initcall' during boot
- Usage :
  - Simply add `'initcall_debug'` to kernel command line
- Result will be available in `dmesg`

initcall_debug

Embedded Experts dedicated to the success of your design

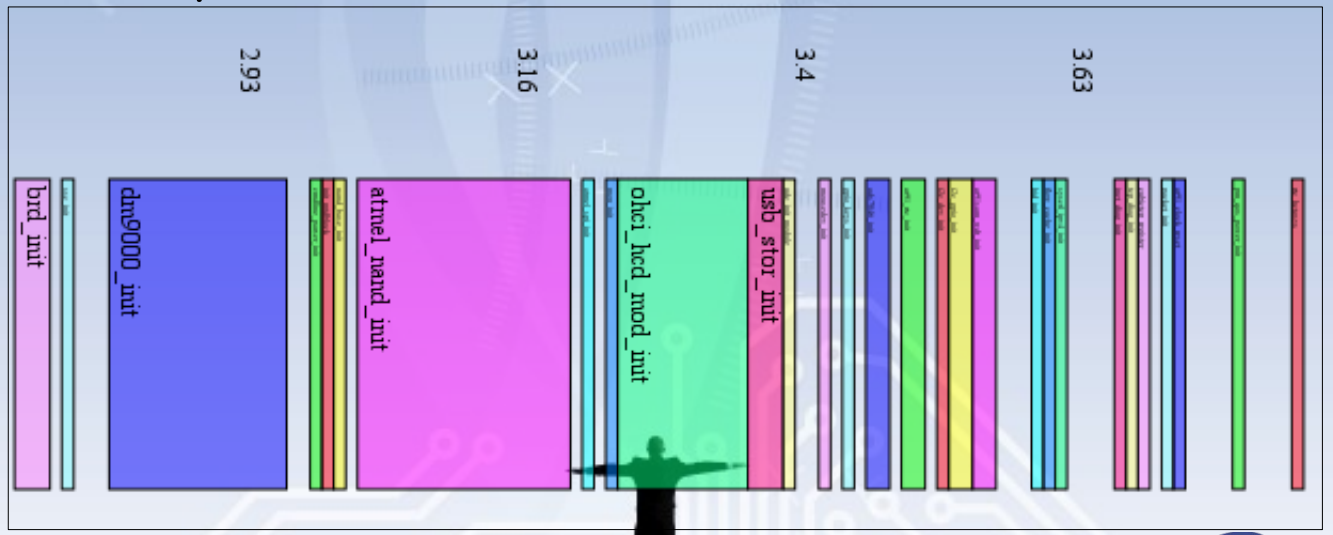
- Exemple of output :
  - `dmesg | egrep 'initcall|calling'`

```
[    1.020000] calling  chr_dev_init+0x0/0xbc @ 1
[    1.030000] initcall chr_dev_init+0x0/0xbc returned 0 after 5047 usecs
[    1.030000] calling  firmware_class_init+0x0/0x78 @ 1
[    1.040000] initcall firmware_class_init+0x0/0x78 returned 0 after 495 usecs
[    1.040000] calling  sysctl_core_init+0x0/0x44 @ 1
[    1.050000] initcall sysctl_core_init+0x0/0x44 returned 0 after 38 usecs
[    1.060000] calling  inet_init+0x0/0x1dc @ 1
[    1.100000] initcall inet_init+0x0/0x1dc returned 0 after 35788 usecs
[    1.100000] calling  af_unix_init+0x0/0x5c @ 1
[    1.110000] initcall af_unix_init+0x0/0x5c returned 0 after 4388 usecs
[    1.120000] calling  populate_rootfs+0x0/0x210 @ 1
[    1.120000] initcall populate_rootfs+0x0/0x210 returned 0 after 495 usecs
[    1.130000] calling  timer_init_sysfs+0x0/0x40 @ 1
[    1.140000] initcall timer_init_sysfs+0x0/0x40 returned 0 after 920 usecs
[    1.140000] calling  fpe_init+0x0/0x84 @ 1
[    1.150000] initcall fpe_init+0x0/0x84 returned 0 after 6369 usecs
```

initcall_debug

Embedded Experts dedicated to the success of your design

- Possibility to change it into a pretty readable graph thanks to Arjan van de Ven's script :
  - `dmesg | perl scripts/bootgraph.pl > boot.svg`



initcall_debug

Embedded Experts dedicated to the success of your design

- Notes about initcall_debug :
  - Increase significantly boot time
  - Log buffer could be too small
    - Increase it size by changing `CONFIG_LOG_BUF_SHIFT` value in the `.config`
- Printk-time must be enabled to get the graph

Time measurement

Embedded Experts dedicated to the success of your design

User-space measurement

Adetel Group

Adeneo Embedded

Bootchart-lite

Embedded Experts dedicated to the success of your design

- http://code.google.com/p/bootchart-lite
- Similar to Bootchart
  - http://www.bootchart.org
- Collects datas from `/proc`
- Written in C
  - Less overhead than a shell script
- Use the same Java parser than Bootchart to change the logs into a nice graph

Bootchart-lite

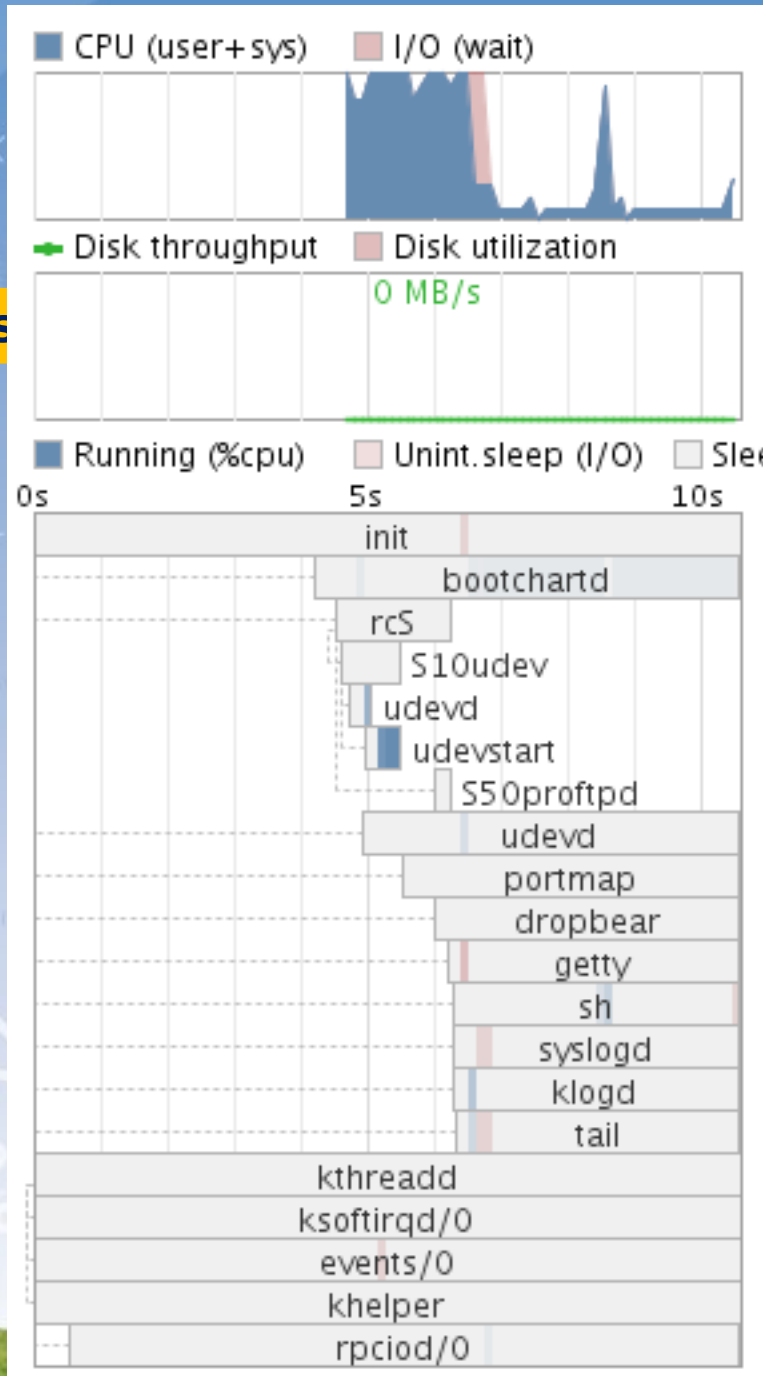
Embedded Experts dedicated to the success of your design

- Usage :
  - Launch bootchart-lite instead of traditional `/sbin/init`
  - Make a gzipped tarball of logs
    - `tar -cvzf bootchart.tgz /etc/bootchart-lite/*`
  - Create the graph using bootchart's java parser :
    - `java -jar ./bootchart.jar bootchart.tgz`

Bootchart-lite

Embedded Experts dedicated to the s

- Bootchart exemple :



Adetel Group

Bootchart-lite

Embedded Experts dedicated to the success of your design

- Notes about bootchart-lite :
  - The bootchart daemon could be stopped manually :
    - Send `SIGUSR1` to bootchartd
  - Or automatically :
    - `#define EXIT_PROC "proc_name"`
  - Can easily modify the sample time

Embedded Experts dedicated to the success of your design

Techniques for optimization

Adetel Group

Adeneo Embedded

Techniques for optimizations

Embedded Experts dedicated to the success of your design

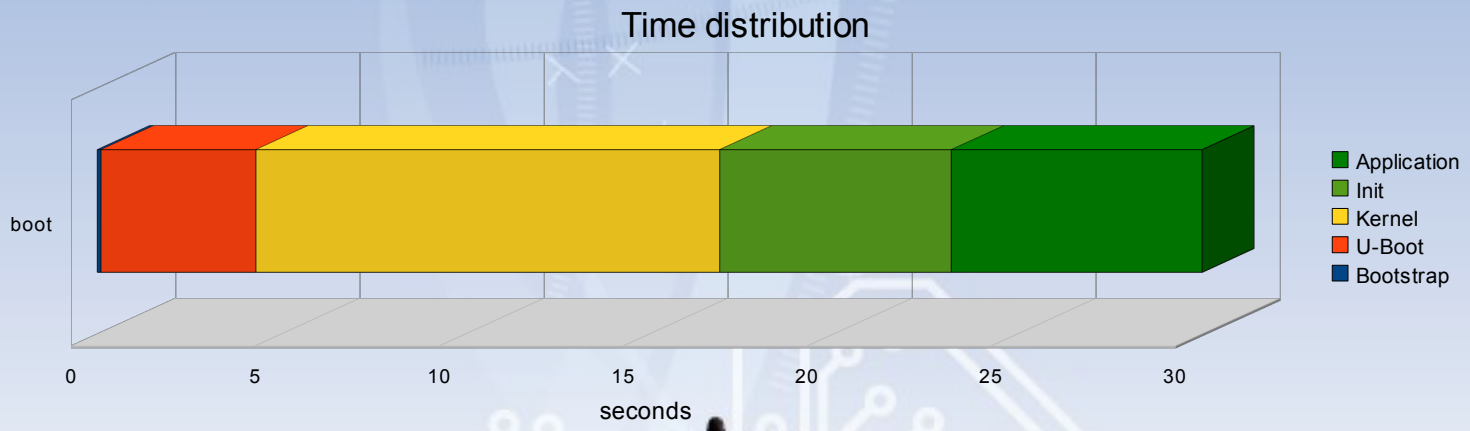
- Our environment
  - Atmel AT91SAM9261-EK board
  - 2.6.29 Kernel
  - U-Boot v2009.06
  - at91sam9261ek_defconfig
  - JFFS2 filesystem (on 122MB partition)
  - Qt for embedded Linux graphic application
    - Start **30 seconds** after the board have been powered up

Techniques for optimizations

Embedded Experts dedicated to the success of your design

- Initial timing :

Bootstrap	U-Boot	Kernel	User-space	
			Init	application
0.1	4.2	12.6	6.3	6.8



Adetel Group

Adeneo Embedded

Techniques for optimizations

Embedded Experts dedicated to the success of your design

User-Space optimizations

Adetel Group

Adeneo Embedded

User-Space optimizations

Embedded Experts dedicated to the success of your design

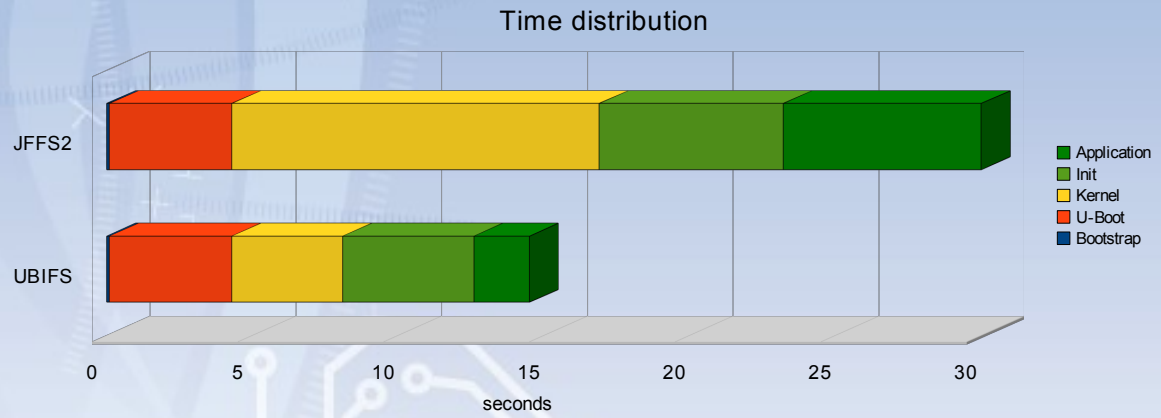
- Use UBIFS instead JFFS2
  - Much faster for mounting, writing, reading
  - Will improve both kernel and user-space initialization time
  - User-space applications will be loaded faster

User-Space optimizations

Embedded Experts dedicated to the success of your design

- Results
  - Boot time : **14.5s**
  - Saving : **15.5s**

Bootstrap	U-Boot	Kernel	User-space	
			Init	application
0.1	4.2	3.8	4.5	1.9



Adetel Group

Adeneo Embedded

User-Space optimizations

Embedded Experts dedicated to the success of your design

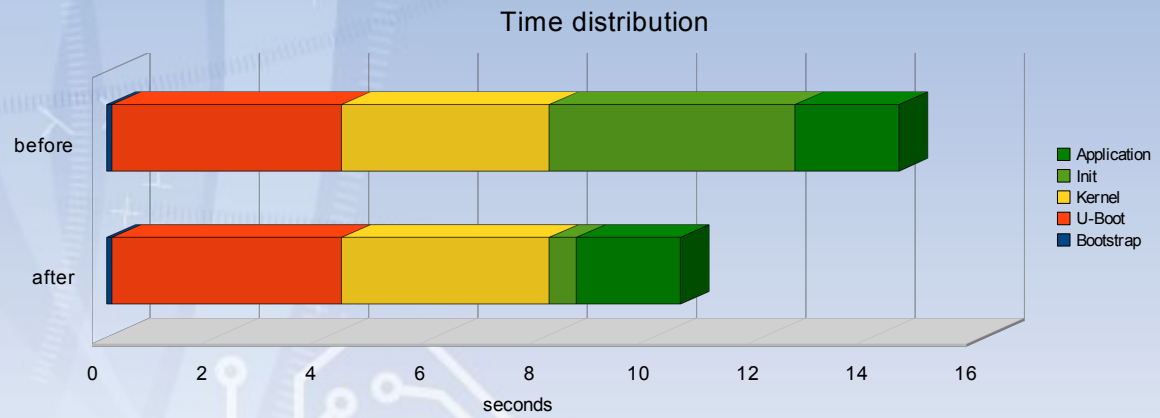
- Remove unneeded rc scripts
  - For example : No need to start `udev` daemon (hardware configuration in embedded systems is often constant)
  - Keep only the scripts which are essential for your application
  - Still possible to start the other scripts later

User-Space optimizations

Embedded Experts dedicated to the success of your design

- Results
  - Boot time : **10.5s**
  - Saving : **4s**

Bootstrap	U-Boot	Kernel	User-space	
			Init	application
0.1	4.2	3.8	0.5	1.9



Adetel Group

Adeneo Embedded

User-Space optimizations

Embedded Experts dedicated to the success of your design

- Dynamically linked applications take a long time to get loaded
- Use statically linked application

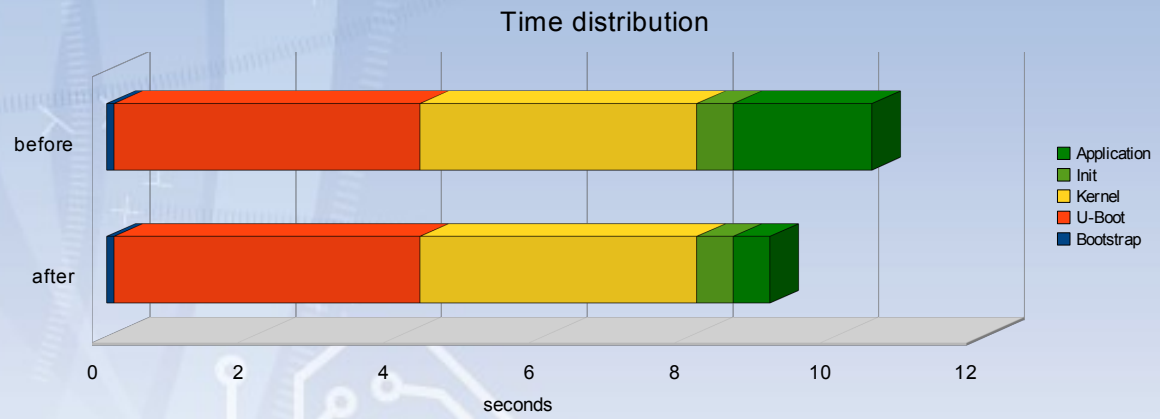
  Note : The binary size can increase significantly
  - In our example :
    - Dinamically linked : 152KB
    - Statically linked : 12MB

User-Space optimizations

Embedded Experts dedicated to the success of your design

- Results
  - Boot time : **9.1s**
  - Saving : **1.4s**

Bootstrap	U-Boot	Kernel	User-space	
			Init	application
0.1	4.2	3.8	0.5	0.5



Adetel Group

Adeneo Embedded

User-Space optimizations

Embedded Experts dedicated to the success of your design

- Other solutions for optimisation
  - /sbin/init alternatives
    - Initng
    - Upstart
      - Not suitable for small systems
      - Upstart needs additionnal libs such as Dbus
- Better to optimize init process "by hand"

Techniques for optimizations

Embedded Experts dedicated to the success of your design

Kernel optimizations

Adetel Group

Adeneo Embedded

Kernel optimizations

Embedded Experts dedicated to the success of your design

- Identify what is essential for your system just after boot
- Remove unused features
- Use deferred modules
  - Initialize modules after the system completely booted, by doing

```
echo 1 >/proc/deferred_initcalls
```

Kernel optimizations

Embedded Experts dedicated to the success of your design

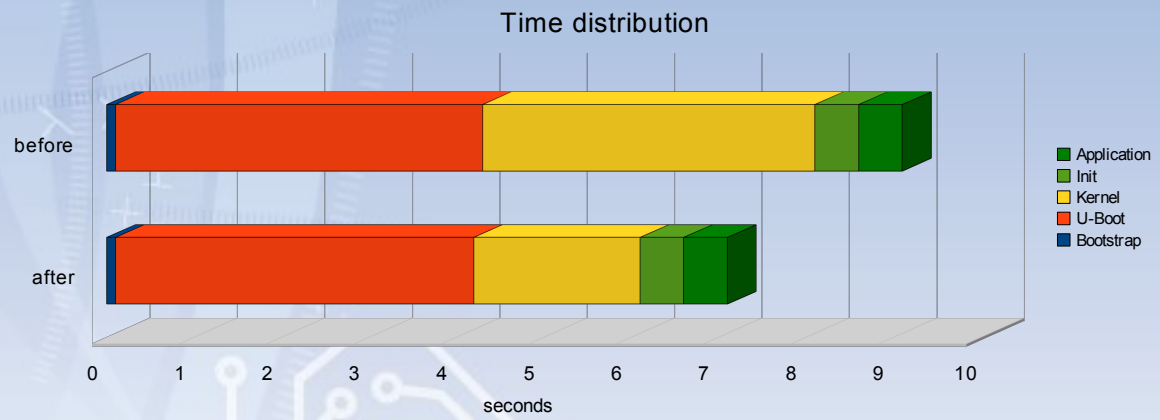
- Best strategy to configure the kernel ?
  - Don't use '`make yourboard_defconfig`'
  - Do '`make allnoconfig`'
  - Then select the minimum required by your system

Kernel optimizations

Embedded Experts dedicated to the success of your design

- Results
  - Boot time : **7.1s**
  - Saving : **2s**

Bootstrap	U-Boot	Kernel	User-space	
			Init	application
0.1	4.1	1.9	0.5	0.5



Adetel Group

Adeneo Embedded

Kernel optimizations

Embedded Experts dedicated to the success of your design

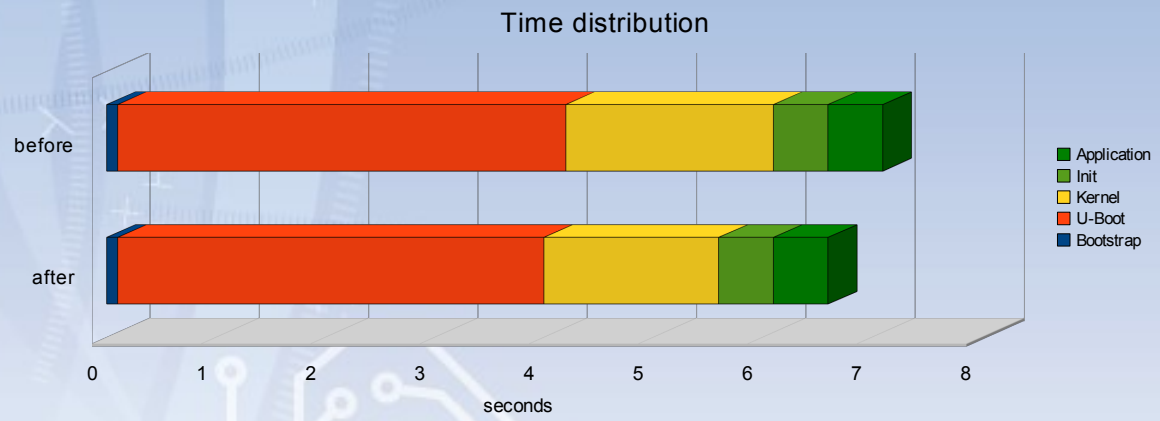
- Set 'CONFIG_EMBEDDED=y'
- [*]Configure standard kernel features
  - Allow disabling more (unused) features in the kernel
  - Reduce the kernel size

Kernel optimizations

Embedded Experts dedicated to the success of your design

- Results
  - Boot time : **6.6s**
  - Saving : **500ms**

Bootstrap	U-Boot	Kernel	User-space	
			Init	application
0.1	3.9	1.6	0.5	0.5



Adetel Group

Adeneo Embedded

Kernel optimizations

Embedded Experts dedicated to the success of your design

- Make the kernel boot quietly
  - Disable the printk messages on the console during the kernel boot.
- Usage
  - Simply add `'quiet'` in the kernel command line
    - Ex:

```
root=/dev/mtdblock1 console=ttyS0,115200 rootfstype=jffs2 quiet
```

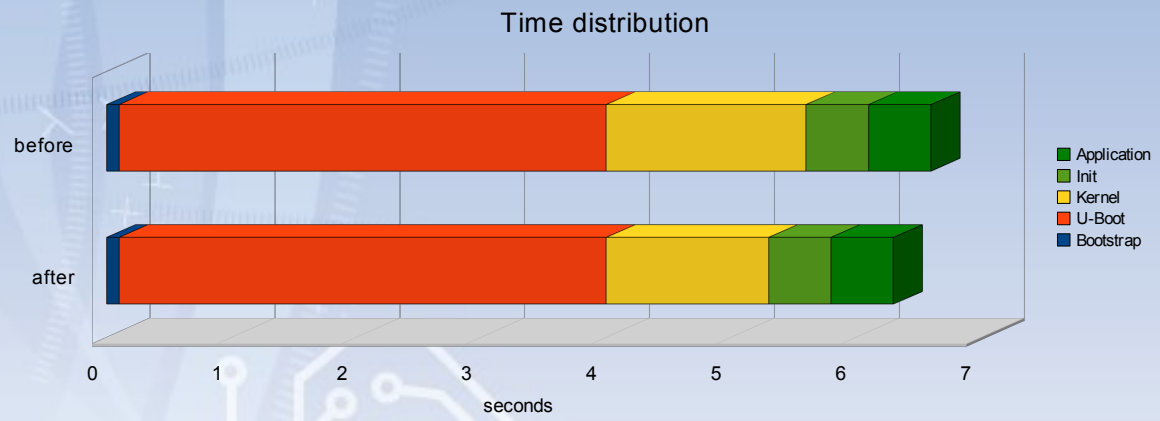
- Time saved will be proportional to number lines printed on the console during boot
- Still possible to consult the boot log in dmesg

Kernel optimizations

Embedded Experts dedicated to the success of your design

- Results
  - Boot time : **6.3s**
  - Saving : **300ms**

Bootstrap	U-Boot	Kernel	User-space	
			Init	application
0.1	3.9	1.3	0.5	0.5



Adetel Group

Adeneo Embedded

Kernel optimizations

Embedded Experts dedicated to the success of your design

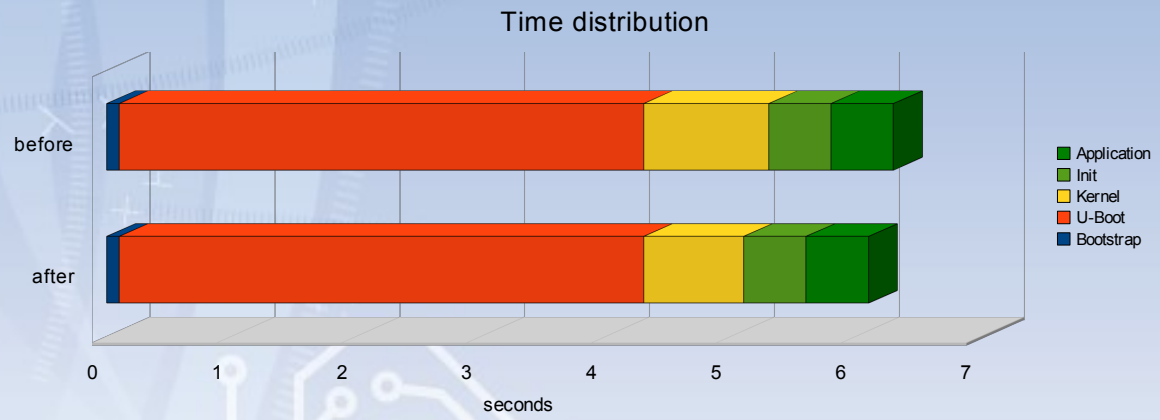
- Preset 'loop per jiffies'
  - Constant value for the architecture, but calculated at each boot
  - Possibility to give this value to the kernel
- Usage :
  - Add `'lpj=<value>'` in the kernel command line

Kernel optimizations

Embedded Experts dedicated to the success of your design

- Results
  - Boot time : **6.1s**
  - Saving : **200ms**

Bootstrap	U-Boot	Kernel	User-space	
			init	application
0.1	3.9	1.1	0.5	0.5



Adetel Group

Adeneo Embedded

Kernel optimizations

Embedded Experts dedicated to the success of your design

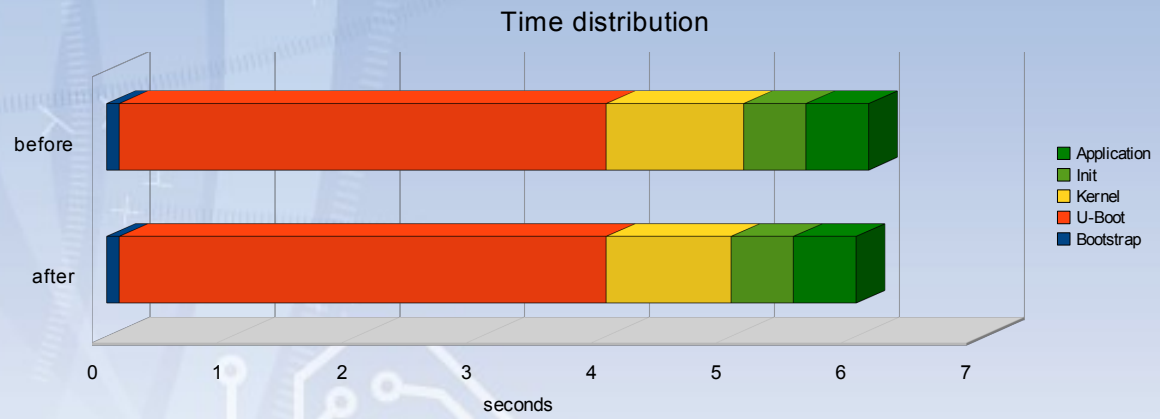
- BBT is created at each boot for NAND devices, and stored in RAM
  - Use On Flash BBT
- We made a patch for Atmel nand driver
  - Now merged into the mainline (2.6.30 kernel)
- Usage
  - Add `'atmel_nand.on_flash_bbt=1'` in the kernel command line

Kernel optimizations

Embedded Experts dedicated to the success of your design

- Results
  - Boot time : **6s**
  - Saving : **100ms**

Bootstrap	U-Boot	Kernel	User-space	
			Init	application
0.1	3.9	1.0	0.5	0.5



Adetel Group

Adeneo Embedded

Kernel optimizations

- Other solutions for optimisation
  - Use new 'fastboot' technologie
    - In the mainline since 2.6.30
    - Allow asynchronous initcalls
    - Usefull if there are CPU and/or disk idle during the kernel boot

Techniques for optimizations

Embedded Experts dedicated to the success of your design

U-Boot optimizations

Adetel Group

Adeneo Embedded

U-Boot optimizations

Embedded Experts dedicated to the success of your design

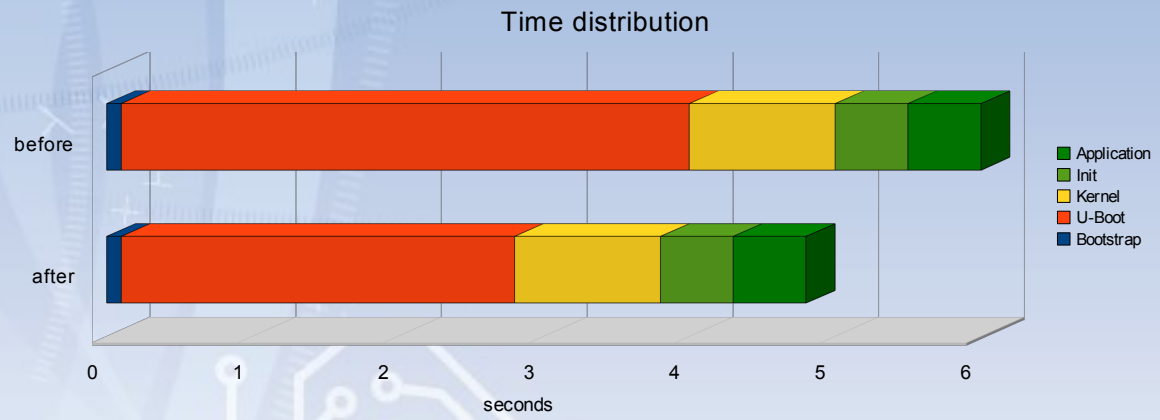
- Customize U-Boot for doing only what is necessary to launch the kernel
- For example in our system we don't need to use these devices in U-Boot
  - Ethernet device
  - Nand device
- We keep LCD display since it consume a very few time (~20ms)

U-Boot optimizations

Embedded Experts dedicated to the success of your design

- Results
  - Boot time : **4.8s**
  - Saving : **1.2s**

Bootstrap	U-Boot	Kernel	User-space	
			Init	application
0.1	2.7	1.0	0.5	0.5



Adetel Group

Adeneo Embedded

Techniques for optimizations

Embedded Experts dedicated to the success of your design

Bootstrap optimizations

Adetel Group

Adeneo Embedded

Bootstrap optimizations

Embedded Experts dedicated to the success of your design

- The bootstrap do some hardware initializations, then launch the kernel
- Official Atmel bootstrap offer two choice for initialize master clock : 100MHz and 125MHz
  - Upgrade to 125MHz will cause the CPU to run faster

Adetel Group

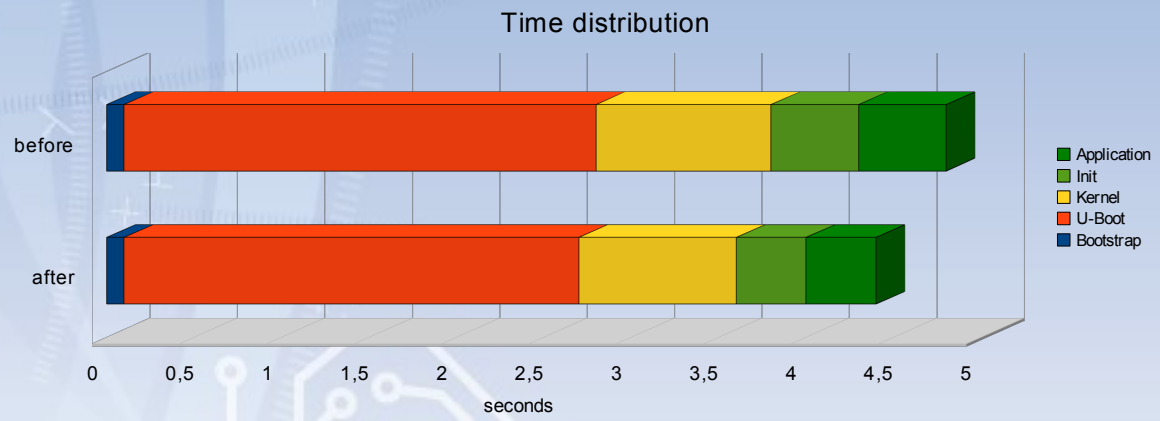
Adeneo Embedded

U-Boot optimizations

Embedded Experts dedicated to the success of your design

- Results
  - Boot time : **4.4s**
  - Saving : **400ms**

Bootstrap	U-Boot	Kernel	User-space	
			Init	application
0.1	2.6	0.9	0.4	0.4



Adetel Group

Adeneo Embedded

Techniques for optimizations

Embedded Experts dedicated to the success of your design

Now, have a look at how much time we saved

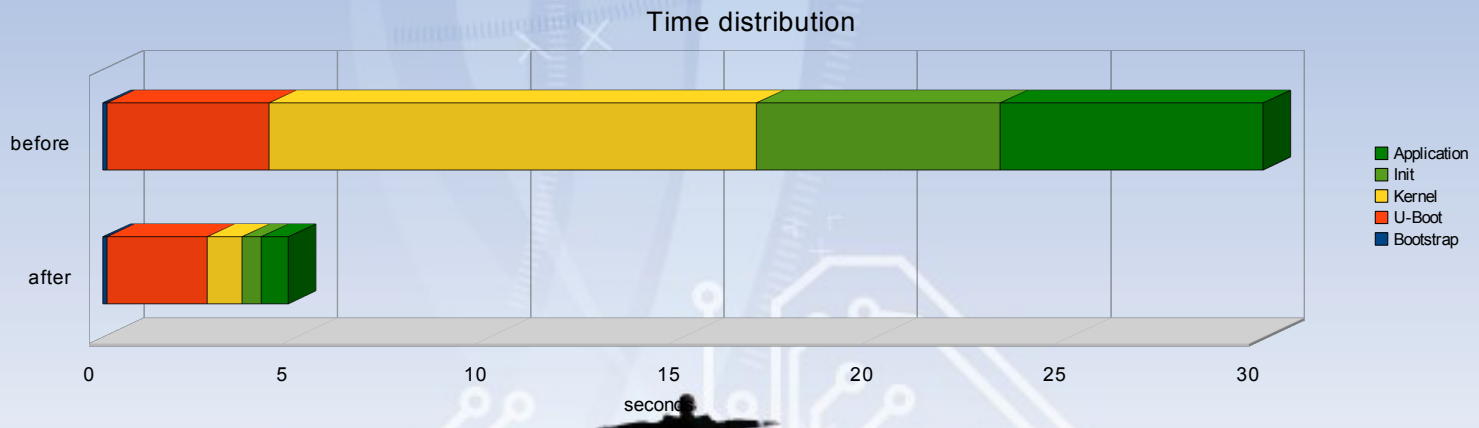
Adetel Group

Adeneo Embedded

Techniques for optimizations

Embedded Experts dedicated to the success of your design

	Bootstrap	U-Boot	Kernel	User-space		Total
				Init	application	
Before optimization	0.1	4.2	12.6	6.3	6.8	30
After optimization	0.1	2.6	0.9	0.4	0.4	4.4



Adetel Group

Adeneo Embedded