INSTITUT FÜR MIKROELEKTRONIK-
UND MECHATRONIK-SYSTEME gGMBH

IMMS

Quantitative analysis of system initialization in embedded Linux systems

André Puschmann

IMMS gGmbH
Ehrenbergstrasse 27
D-98693 Ilmenau, Germany
http://www.imms.de

Tel.: +49 3677-69 55 00
Fax.: +49 3677-69 55 15

E-Mail: andre.puschmann@imms.de

Introduction - facts and figures

- Ilmenau
  - Situated in the south of Thuringia
  - Approx. 30.000 inhabitants

- Ilmenau University of Technology
  - Founded in 1894
  - Approx. 7.000 students
- Institut for Microelectronic and Mechatronic Systems
  - Founded in 1995
  - „Associated Institute of Ilmenau University of Technology“
  - In time 64 employees / around 25 students

Table Of Contents

- Introduction
  - Motivation and goals
  - The boot process
  - Instrumentation and reference system
- Boot time reduction
  - Universal techniques
  - Bootloader layer
  - Kernel layer
  - Application layer
- Conclusion
  - Results and outlook

Motivation

- Linux is widely used, e.g.:
  - Consumer electronic devices
  - Automation
- System's size increases constantly → startup-time too
- But: time is critical

Consider a car audio system which needs 5 min for start up.

Would you like to buy or use it?

Goals

- Identify most critical and time consuming sections
  - But: ensure compliance with “Pareto principle” (80-20 rule)
- Survey techniques/proposals for boot time reduction
- Apply on Gumstix verdex XM4 board
- Evaluate promised and achieved savings

The Boot Process

"Time from power-on to usable system"

(Klahn, Muhammad, ELC2006)

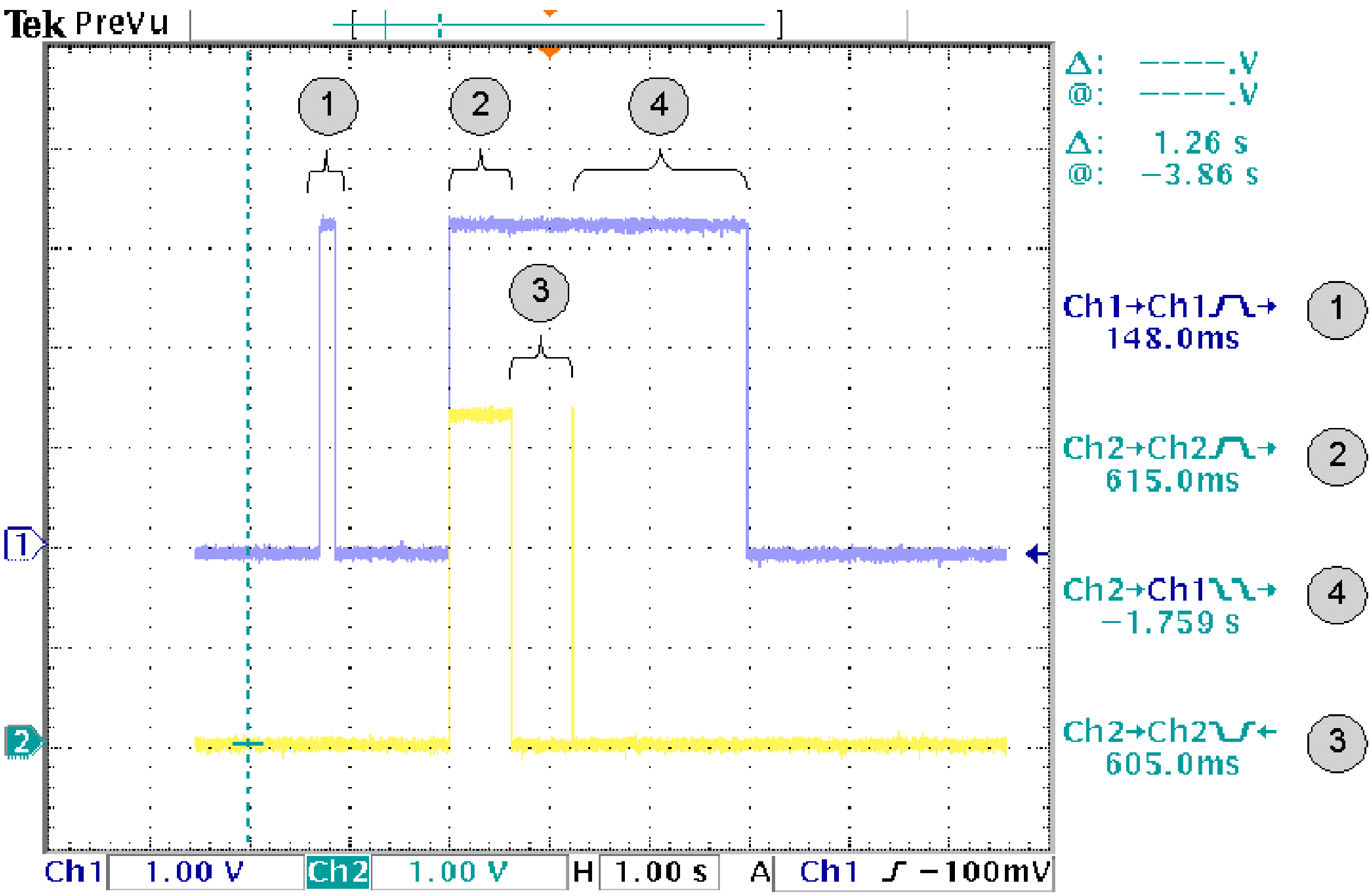
Phases:

- Bootloader phase:
  - Initial hardware setup
  - Load "more complicated system"
- Kernel phase:
  - Most intrinsic part
  - Initialize all hard- and software components
- Application phase:
  - Userspace initialization
  - Longest phase?

Instrumentation (1) - Overview

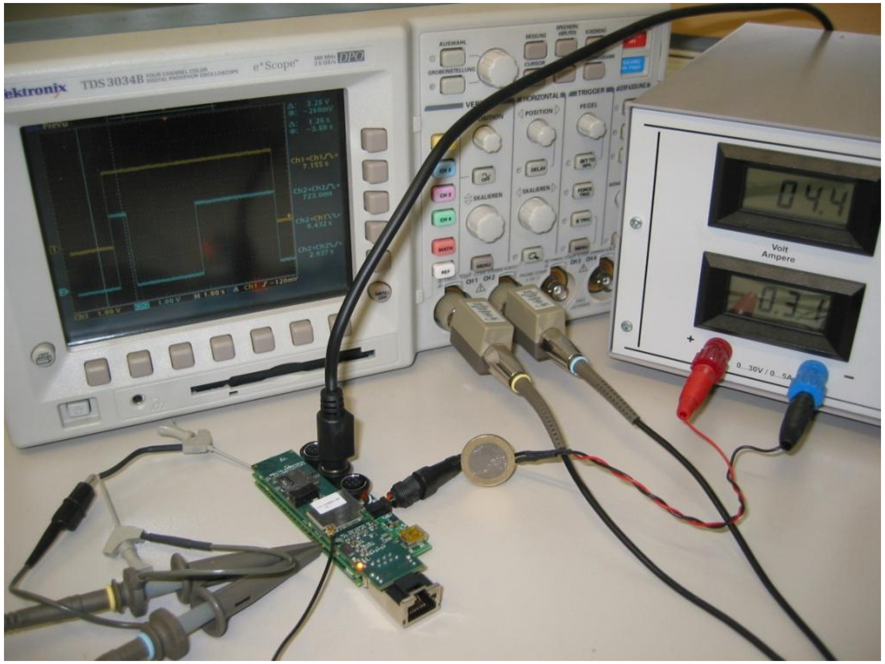
- What we haven't done:
  - Kernelspace measurement (e.g. `printk` times, ..)
  - Userspace measurement (e.g. bootchart, ..)
- What we have done:
  - Each boot phase divided into one or more sections
  - Software based test points that toggle GPIO pins
  - Oscilloscope to measure time-span between logical values

Instrumentation (2) – Example



System Setup

- Hardware:
  - *Gumstix Verdex XM4* mainboard
  - Marvel XScale PXA270 at 400MHz (ARM v5TE)
  - Ethernet extension board
  - 64 MB RAM
  - 16 MB Intel NOR flash



- Software:
  - gumstix-buildroot
  - U-Boot 1.2
  - Linux Kernel 2.6.21

Boot Time Reduction - Overview

"Ideal world of 100% disk and CPU utilization"

(Ziga Mahkovec, bootchart.org)

Layer oriented:

- Universal techniques
- Bootloader layer techniques
- Kernel layer techniques
- Application layer techniques

Universal Techniques (1)

- File and code size
  - Board support packages are often full blown with features
  - Only enable features that you really need
- Compression
  - Less space is needed (most often)
  - Not necessarily faster
  - Weigh up speed of storage media and decompression speed

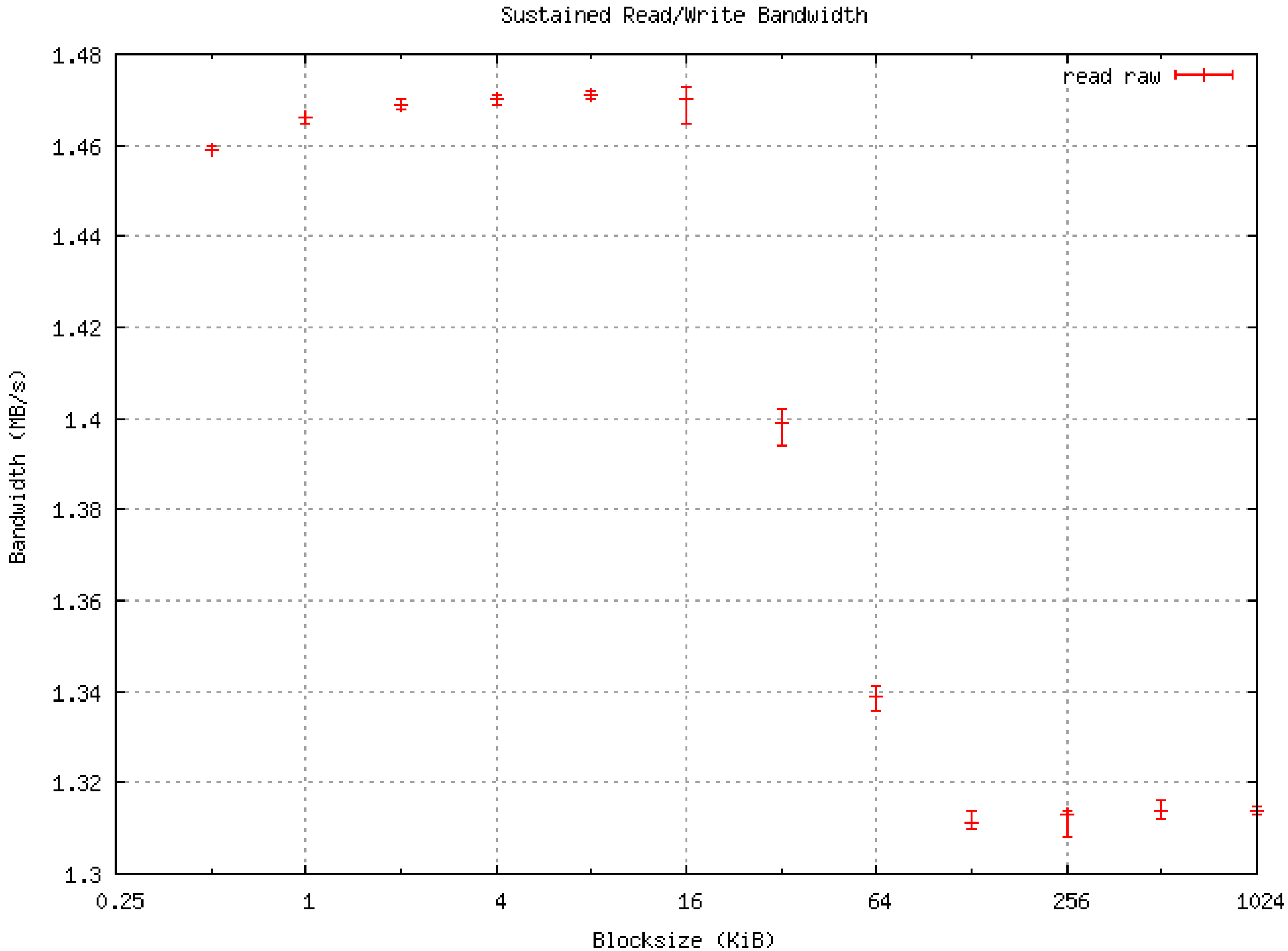
Universal Techniques (2)

- Execution in place (XIP)
  - Execute code directly from non-volatile memory
  - Pros:
    - Lower time-to-start
    - Less RAM and power consumption
  - Cons:
    - Needs more (expensive NOR) flash
    - Possible lower overall throughput

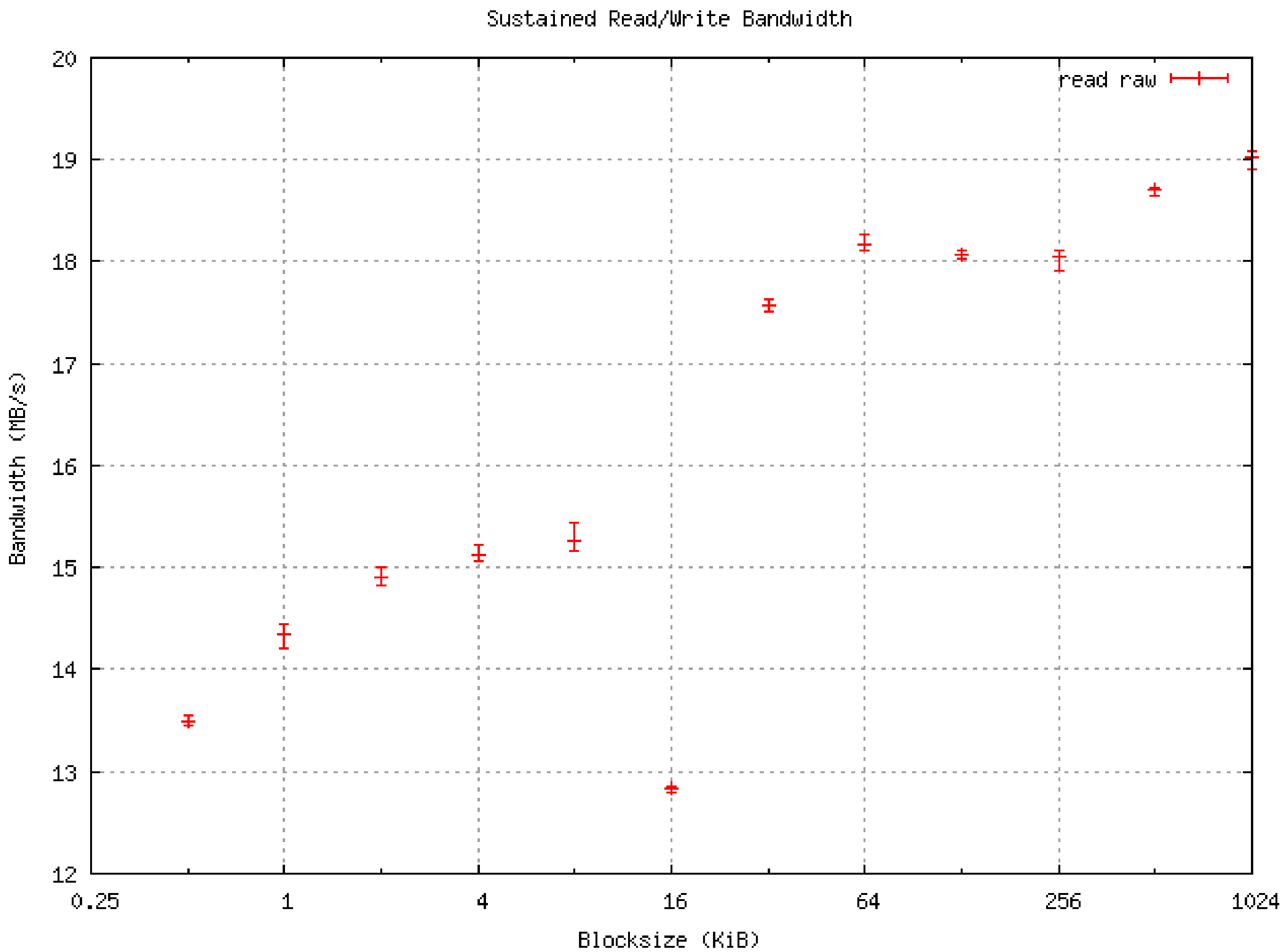
Universal Techniques (3)

- Memory configuration
  - RAM is much faster than flash → flash is bottleneck
  - Fully use the capabilities of the memory chips
    - → Review memory (timing) configuration
    - → Review cache configuration
- Flash read performance analysis
  - GNU tools, e.g. `time cp /dev/mtd2 /dev/null` or `dd`
  - *flanatoo* ("A Flash Analysis Tool")

Flash Read Performance (1) - Original



Flash Read Performance (2) - Optimized



Bootloader Layer Techniques - Effort Justified?

- Bootloader phase only of relatively short duration
- Optimization quite complex
- Promises only marginal improvement
- But: General rules also apply here
- Don't spent too much time, unless there is a good reason

- Little orientation (u-boot startup):
  - unoptimized: 150 ms
  - optimized: 135 ms

Kernel Layer Techniques (1) – Kernel XIP

- Kernel image type comparison:

	normal (non-XIP)	**XIP kernel**	**ZBOOT_ROM kernel**
size	885.76 kB	1,826.63 kB	885.76 kB
copy/decompress	730 ms	65 ms	616 ms
kernel init	275 ms	535 ms	275 ms
mount/start root fs	679 ms	1,156 ms	679 ms
total	1,684 ms	1,756 ms	1,570 ms

Kernel Layer Techniques (2)

- Eliminate console output:
  - Append `quiet` parameter to kernel command line
  - Reduction of fairly 300 ms
- Eliminate lpj calculation:
  - Append calculated `lpj` parameter to kernel command line
  - Reduction of 200 ms
- Driver initialization:
  - Avoid firmware loading/hardware probing overhead
  - Deferred and concurrent driver initialization with modules
  - Improvement driver specific

Application Layer Techniques (1) – Application XIP

- Filesystem comparison:
  1. CramFS, access through mtd layer
  2. Linear CramFS, XIP disabled
  3. Linear CramFS, busybox marked as XIP
  4. AXFS, no XIP, profiling off

	1	**2**	**3**	**4**
mount filesystem	0.016 s	0.024 s	0.024 s	0.012 s
start user scripts	1.119 s	0.892 s	0.935 s	1.019 s
start demo application	0.140 s	0.100 s	0.150 s	0.110 s
copy dummy file (512 kB)	0.150 s	0.110 s	0.060 s	0.050 s
total	1.425 s	1.126 s	1.169 s	1.191 s

Application Layer Techniques (2)

- Device node population
  - *udev/mdev* increase system comfort, quality and start up time
  - Way out: copy static nodes, start hotplug-daemon afterwards
  - Reduction of 200 ms

- Parallel init script execution
  - Replace *init* with *myinit*
  - Allows parallel execution and dependencies
  - Potential savings application specific
  - Here: Reduction of 200 ms

Conclusion (1) - Results

- Use thin system configuration
- Try to avoid udev at system startup
- Review memory configuration
- Enable caches as soon as possible
- Evaluate benefit of compression and XIP

	partly optimized	optimized
bootloader phase	152 ms	135 ms
kernel copy/decompression	814 ms	600 ms
kernel initialization	1,343 ms	247 ms
application phase	1,047 ms	488 ms
total	3,356 ms	1,470 ms

Conclusion (2)

- What have we done:
  - Survey several techniques for boot time reduction
  - Evaluation on widely used PXA270 hardware

- Outlook
  - Further research necessary
  - Things on our To-do list
    - Impact of techniques like prelinking and library optimization
    - Flash read performance (synchronous operation mode)
    - *AXFS* with XIP enabled

The End

Questions, comments?!?