CHEMNITZ UNIVERSITY
OF TECHNOLOGY

Faculty of Computer Science
Real-Time Group

Diploma Thesis

An Analysis of Linux Boot Times

Author: Florian Strunk
Author e-mail: florian.strunk@s2001.tu-chemnitz.de

Date of Issue: September 13, 2007
Date of Submission: March 10, 2008
Supervisor: Dr.-Ing. Robert Baumgartl

Florian Strunk

An Analysis of Linux Boot Times

Diploma Thesis, Chemnitz University of Technology, 2008

Abstract

For consumer devices it is necessary that they are in a ready to use state approximately in a few of seconds (10 -15 sec.) after power on. So far one main objection against the usage of Linux for consumer devices is its time-consuming boot process.
In this thesis it is examined, what the history of the boot times of Linux in the last years is. For this purpose some of the popularly distributions in the versions of the last two years are evaluated. Besides the influence of several mass storages (local and network) and different number of cores of the CPU is shown. In this thesis a possibility is presented, which accelerates the starting up of Linux up to 60 percent compared to the new init-system upstart. For decreasing the time to get in the ready to use state several optimizations are recommended.
Because of the reason that the operating system is not only responsible for the total boot up time, the execution of several firmwares are examined too.

Thanks

I would like to thank Dr.-Ing. Robert Baumgartl for the supervision of this diploma thesis and Nigel Cunningham for the help to let me understand his project TuxOnIce.

Overview of contents

Contents

List of figures

List of tables

1 Introduction

In the past, at the beginning of the computer era, a single computer occupied a whole floor or building. Nowadays thousands of nodes are situated within a separate building to form a huge electronic data processing center. At this time, a computer was used by many persons at the same time and it was turned on by an engineer in the morning before the beginning of the work by employees, in case the computer was turned off at all.

Instead of using one device all day long, the trend goes to utilize a couple of gadgets, which are interconnected to each other. A gadget has at least one primary function, and dozens of secondary functions, to be competitive. For this a operating system is necessary, which can manage these complex structures. The development of a proprietary operating system is very time consuming and expensive. As an alternative the open source operating system Linux can be used. Everyone has the possibility to fit it to the needed requirements. Linux runs on embedded systems with limitation of computing power, up to the TOP500, the fastest 500 computers of the world. From the TOP500 426 of the use Linux as operating system (stand of Nov. 2007) [TOP07].

A survey conducted by Venture Development Corporation (VDC) for the year 2007 says, that 40 percent of the developer use Linux for the running projects as embedded operating system 39 percent are based on commercial and 7 percent on in house developments. In the next project, 87 percent of the embedded system developers want to use Linux, 12 percent a commercial operating system and only 1 percent a in house solution. [LMA07a]

The most people have contact with Linux, but most of these do not know it at all. Which devices are powered by Linux? For example there are low-cost-router, mobile phones, personal video recorders (PVR), electronic devices for kids (like ASUS EEE-PC), personal computers, server, mainframes.

For a device, which is frequently used, but only for a short time span, it is necessary that the device gets in ready state as soon as possible when it is powered on. When the resource energy is unlimited and for free, the device can run 24 hours a day, so that it is ready to use every time. But in most cases the energy is neither unlimited nor free. This means that the device has to be turned on and booted up first, before it can be used. The problem is, that a system, which provides a couple of functions, needs more than a couple of seconds, usually about a minute to finish setup up. But who wants to wait a minute or more when using the DVD player or PVR and who wants to wait more than a couple of seconds when looking for emails? The least of us. Many more examples could be found.

To reduce the time of waiting, desirable are 15 seconds and less, it is appropriate to examine the boot process of Linux. The questions are, how does the boot process function, what evolution took place in the last years, what is the influence of several types of mass storages, how can the boot process be optimized, are there other possibilities to get the system in a ready to use state in a couple of seconds?

It is not the ambition to reduce a system to such a degree that it becomes unusable. A system which could achieve the shortest booting up time, would grabs the headlines, but is likely to fail functionality and usability.
Potential optimizations or changes should not have any adverse effects on compatibility, safety and usability.

2 State of the art

2.1 Overview of the boot process

Let us take a look at a possible boot process of a x86-PC with Linux as operating system. The boot process can be subdivided into four sections.

1. Power on, CPU reset
2. Execute firmware
3. Run boot loader
4. Start operating system
    - Kernel
    - Init-System

For most of the parts, there are several possibilities to perform the necessary function. These are described in chapter 2.2.

2.2 Description of the boot process

2.2.1 Power on, CPU reset

When the computer is powered on, the CPU will be reset and go into a well-definied state. The register will be set to a default value.

2.2.2 Execution of firmware

After the initialization the x86-CPU jumps to the CPU reset vector address and then executes the firmware code (PC-BIOS, LinuxBios, EFI etc.).

2.2.2.1 PC-BIOS

A PC-BIOS (Personal Computer - **Basic Input Output System**) executes the Power On Self Test (POST). The POST can differ from manufacturer to manufacturer (Ami, Phoenix, Award) and from version to version, because no standard does exist. An example of the POST process can be seen in table 2.1.

Keyword	Short description
CPU	set, verify, reset, error flags of CPU
Initialize Support Chips	disconnect Video, Parity DMA and NMI, and initialize the PIT, PIC and DMA chips
Refresh	check capability to refresh PIT chips
Initialize Keyboard	initialize the controller of keyboard and keyboard
ROM BIOS Test	generate checksum of BIOS data and compare result with presetting
CMOS Test	check CMOS chip
Memory Test	check the first 356Ki of memory with routines of the chip set
Cache Initialization	activate external cache
Initialize Vector Table	initialize interrupt vectors and install interrupt table in lower memory
CMOS RAM Test	generate checksum of CMOS RAM, if failure then load defaults
Keyboard Initialization	initialize keyboard, set NumLock to on
Video Test	check and initialize monochrome and CGA graphic interfaces
Video Memory	check video memory of monochrome and CGA graphic interfaces
DMA Test	check DMA controller and page register
PIC Tests	generate some tests for 8259 PIC chips
EISA Mode Test	generate checksum of extended CMOS data, where the information of the EISA interface is placed
Enable Slots	if the test before is positive, the slots 0 – 15 will be enabled
Memory Size	write test to all addresses above 256Ki with 64Ki blocks and init them. If one bit of a block faulty, then this block and all blocks above won't be seen.
Memory Test	write and read test of the found size above 256Ki
EISA Memory	initialize all EISA slots and check memory of EISA interfaces
Mouse Initialization	searching for the input device mouse and installing interrupt vector
Cache Initialization	initialize cache controller
Shadow RAM Setup	enable all shadow ram, which are activated by CMOS setup
Floppy Test	check and initialize floppy controller and drive
Hard Drive Test	check and initialize hard disk controller and drive
Serial/Parallel	initialize all serial and parallel ports, use the I/O port information of the CMOS setup to find the ports.

Math coprocessor	initialize the math coprocessor
Boot Speed	set the default core speed
Manufacturing POST Loop	reboot if loop pin set
Security Check	ask user for password, if one was set
Write CMOS	write CMOS setup data to ram
Pre boot	waiting for previous process
Adapter ROM Initialization	initialize all ROMs, which are found between C800:0 and EFFF:F each ROM checks and initializes all founded devices
Setup Time	set the CMOS time to the entry of the data from address 40 h
Boot System	give the control of int 19 to boot loader
Boot Errors	if the boot loader fails, the BIOS will try to boot from floppy drive

Table 2.1: Post process of an Award PC-BIOS version 4.53, source [SCH07]

2.2.2.2 LinuxBios

The idea of the LinuxBios is besides providing an open firmware reducing the components which will be initialized . It only initializes those components, witch are necessary to load a Linux kernel. However not many main boards actually have enough flash ROM capacity for a kernel, so that an other payload can be loaded before. Several payloads are listed in table 2.2.

Tasks executed by the LinuxBios in detail (confer source [MIN04]):

- The first 10 to 15 instructions initializes the CPU, a minimal virtual memory (at minimum 32-bit addressing mode) and other resources that are needed to turn on memory, such as the I2C bus. The internal CPU to also set to a sane state. They also set the internal CPU to a sane state.
- Then the startup code for the memory follows, which needs a sane CPU and a working I2C bus for requesting the memory parameters.
- After that, some code is executed for loading object code originally written in C from the Flash into the main memory. As an option compressed object code can be used.
- Then the code, which needs a working main memory, can be run. This scans the hardware resources and initializes them.
- At last, one or more payloads can be loaded and executed, which perform custom and final configuration work and boot an operating system.

Payload	Comment	URL
Linux	Boot into a Linux kernel directly	http://www.kernel.org/
FILO	Simple bootloader with filesystem support	http://www.linuxbios.org/FILO
GRUB2	Will replace FILO, does not work yet	http://www.linuxbios.org/GRUB2
Mitch Bradley's Open Firmware	IEEE1275-1994 Open Firmware	http://www.openbios.org/Open_Firmware
CodeGen's SmartFirmware	IEEE1275-1994 Open Firmware	http://www.openbios.org/SmartFirmware
OpenBios	IEEE1275-1994 Open Firmware	http://www.linuxbios.org/OpenBIOS
GNUFI	(UEFI)	http://www.gnu.org/software/gnufi/
Etherboot	Includes FILO, and its FILO supports SATA and USB booting	http://www.linuxbios.org/Etherboot
ADLO	Glue layer to 16-bit Bochs BIOS. Allows booting Windows and OpenBSD	http://www.linuxbios.org/ADLO
Plan 9	A distributed operating system	http://www.linuxbios.org/Plan_9
memtest86	Can stress-test your RAM	http://www.linuxbios.org/Memtest86
RedBoot / eCos	Real-time OS for embedded systems; initial port to ELF completed but no longer available.	http://www.linuxbios.org/RedBoot

Table 2.2: Available payloads for the LinuxBios, confer source [COR]

2.2.2.3 (U)EFI

In the mid-1990s, Intel has developed a new platform initialization for the IA64 architecture named EFI (Extensible Firmware Interface). The Version 1.1 of EFI was used as a starting point from the UEFI (Unified EFI) forum.

In contrast to the Linux BIOS, the UEFI gets more control over the hardware. The operating system can only use the specified interfaces. A boot sequence with an EFI is illustrated in figure 2.1.

Some of the features of UEFI are:

- UEFI is not specific towards processor architectures
- provides a processor-independent device driver environment, called EFI Byte Code or EBC
- supports remote maintenance
- supports a GUID Partition Table (GPT)
- supports FAT32 as file system
- supports an EFI shell

- supports a compatibility mode to PC BOIS

The whole specification of UEFI 2.1 can be downloaded from [UEFI2.1] after subscribing an agreement.

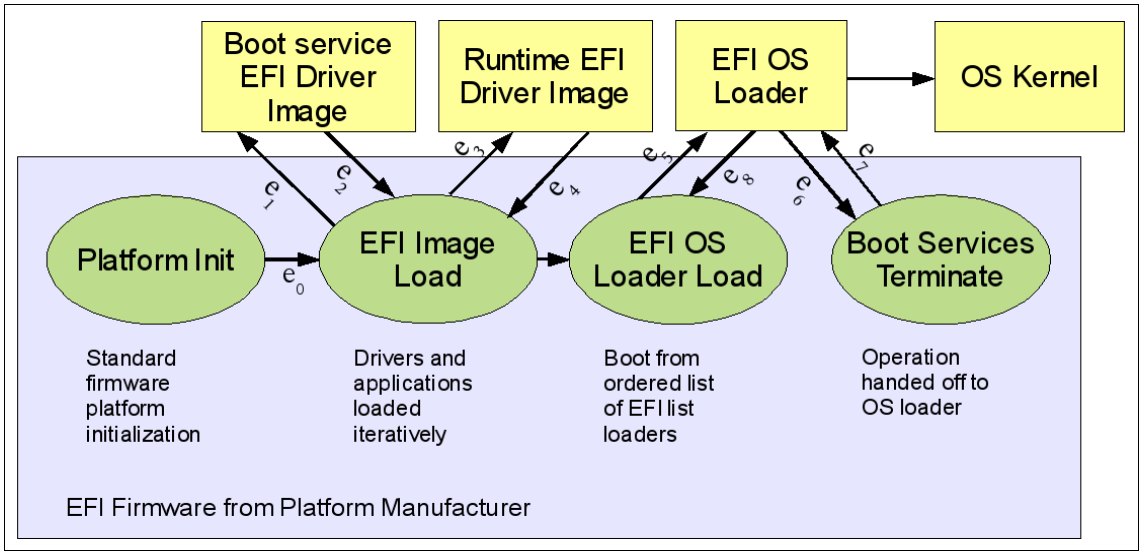


Figure 2.1: Platform and EFI OS Booting Sequence, source [UEFI1.1]

2.2.3 Load boot loader

If no boot loader is included in the firmware, the boot loader will be loaded from the mass storage device. The boot loader can load the initrd or initramfs image into the RAM (or the initrd / initramfs image can be loaded by the kernel later). After that, the kernel image will be loaded and the kernel options will be committed.

2.2.4 Load operation system

2.2.4.1 Kernel

After the kernel is loaded, the hardware detection is executed by the kernel. The supported hardware will be initialized. Next, the network getting started and the file system getting mounted.

2.2.4.2 Init

The init process getting started. First, the init script for the default runlevel is executed. Depending on which distribution is used , several runlevels (confer table 2.3 and 2.4) are running through to the destination runlevel.

A boot sequence of a Linux based operating system is illustrated in figure 2.2.

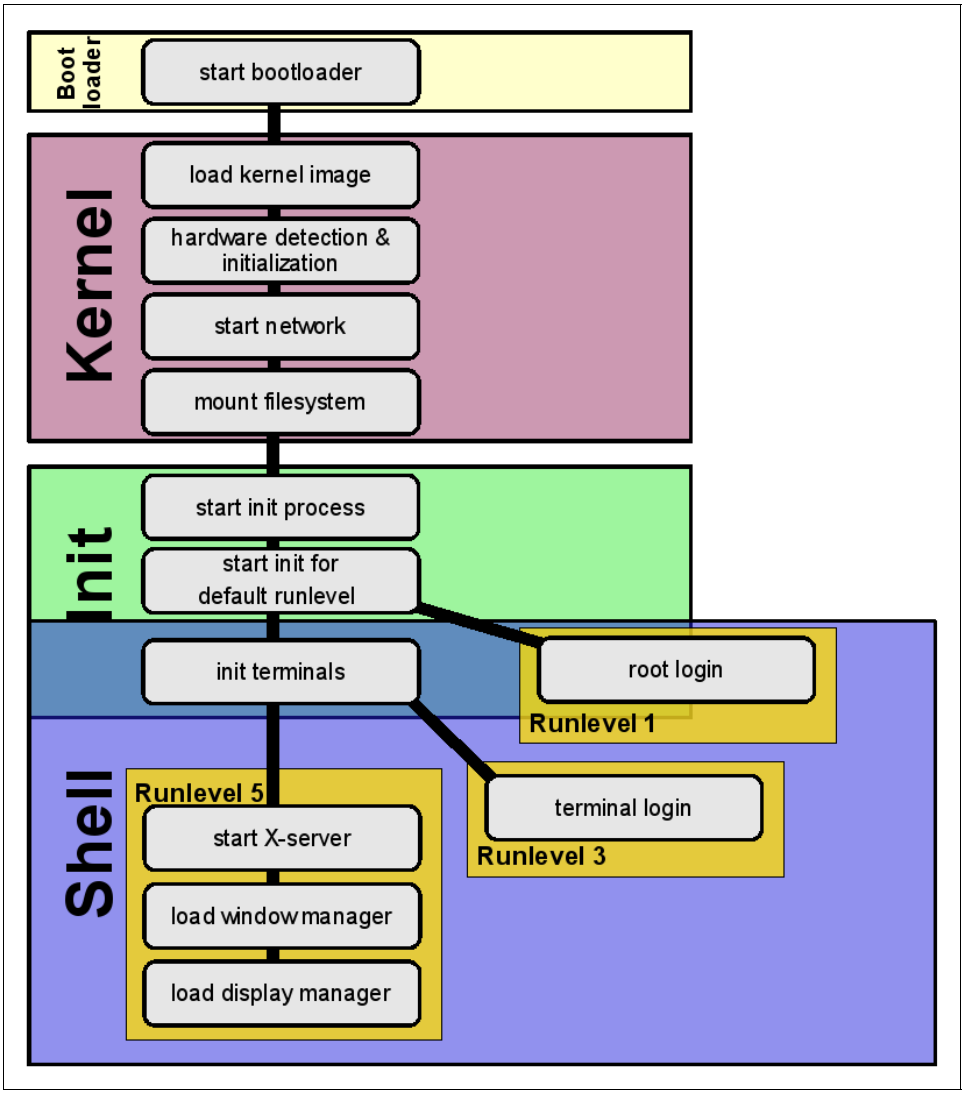


Figure 2.2: Simplified illustration of a Linux boot sequence, confer source [WIKId]

Runlevel	Description
0	Shutdown with Halt
1 and S	Single user
2	Multi user without network, without NFS
3	Multi user with network, but without starting X
4	Usually not used
5	Multi user with network and with starting X
6	Shutdown with reboot

Table 2.3: Runlevel for Fedora, Red Hat and SuSE, confer source [KOF07a]

Runlevel	Description
S	Initialization of the computer immediately after start
0	Shutdown with Halt
1	Single user with network
2-5	Multi user with network and starting X
6	Shutdown with reboot

Table 2.4: Runlevel for Debian and Ubuntu, confer source [KOF07a]

2.3 Previous publications on "Boot Process and Linux"

Many publications on the topic *Boot Process and Linux*. The most important and latest publications are listed below.

The editors of the German journal *Linux Magazin* have compared in the article *Boot-Beschleuniger im Vergleich* (engl. Comparison of accelerator for the boot process) [LMA05] several init system, which should be used as an alternative to the aged System-V-Init. Approximately one year later, a separate article [LMA07b] came out with a research of Upstart, another init system. The student Daniel Parthey has check Initng as substitution of System-V-Init and LinuxBios as an alternative to the PC-BIOS in his student research project [PAR06]. At the Embedded Linux Conference Europe (ELC Europe) in the year 2007, Mr. Vitaly Wool has presented the results of his research on Parallelizing Linux boot on CE Devices [WOO07]. An essential part of the work was to load and initialize drivers in parallel. Besides this fcache exists, which has started as a weekend project of Jens Axboe in the year 2006 [AXB06]. Fcache is a remapping cache that is implemented between the file system and block device. It reduces the seek time of the random read workload of a hard disk by changing it to sequential access. With readahead and other preload mechanisms, many alternative to fcache do exist, which are used in many distributions. At the Ottava Linux Symposium (OLS) in the year 2007, Mr. Michael Opdenacker has presented his project with the theme *Readahead – Time Travel Techniques* [OPD07]. It compares several implementations of readahead versions.

On the openSUSE wiki page there are some articles on the theme of reducing boot time, e. g. the use of fcache under openSUSE [SUS07a] and in another test the use of upstart [SUS07b]. At this test, upstart was used only in the compatibility mode to System-V-Init, so that it could not get any profit out of the new techniques of upstart. For getting the best performance of upstart, specific scripts has to be used.

An disadvantage of all these publications is, that in each case an other computer systems was used. Single core as well as dual core systems with a variety of distinct CPUs and different clock speed have been used. As mass storage several 2.5" and 3.5" hard disks were utilized. An other problem of the most publications is, that there are no specifics of the used components e. g. manufacturers and product names of the hard disks are missing and whether the command queuing is supported by the hard disk and the

controller or not. Because of all this, the results of the several publications can not be compared to each other.
To conclude from all these publications, is it obscure, what influence the distinct components and accelerating techniques have on the total boot time.

3 Measuring methods

3.1 Introduction of the measuring methods

3.1.1 Stop watch

3.1.1.1 Short Description of using a stop watch

The most simple way to get a measurement is to use a stop watch. A disadvantage is, that the accuracy is dependent on the person, that takes the measurement. If the period of a measuring is long in comparison to the reaction time, the reaction time carries no weight.
An advantage is, that this method of measuring does not have a negative effect on the system, which is under examination.

How exact a measurement with a stop watch is, is discussed in chapter 3.2.3.

3.1.1.2 Measuring points

The starting and the end point of a measurement with the stop watch can be selected free. The best choice is to have an acoustic or visual signal at these points.
Unless otherwise noted, the starting point is when typing the return key in the boot loader Grub, that starts the Linux boot process. As end point the ready to use state of the display manager GDM is used. This state is reached as visual signal, when the mouse pointer changes from the clock symbol to an array symbol and as the acoustic signal, when the ready to use sound is played.

3.1.2 Bootchart

3.1.2.1 Short Description of Bootchart

“Bootchart is a tool for performance analysis and visualization of the GNU/Linux boot process. Resource utilization and process information are collected during the boot process and are later rendered in a PNG, SVG or EPS encoded chart.”
“... Bootchart provides a shell script to be run by the kernel in the `init` phase. The script will run in background and collect process information, CPU statistics and disk usage statistics from the `/proc` file system. The performance data are stored in memory and are written to disk once the boot process completes.“
Quotes from [BOOa]

3.1.2.2 Measuring points

The starting point of measuring is equivalent to the starting point of the kernel and the end equals the starting point of the display manager like e. g. XDM, GDM or KDM.

3.1.2.3 How to install Bootchart

- download the `bootchart-0.9.tar.bz2` from [BOOb]
- run the `install.sh` script for installing bootchartd
- add `init=/sbin/bootchartd` to the kernel options in the boot loader file
- the chart can be generated as png-file with the following command
  ```
  curl --form format=png --form log=@/var/log/bootchart.tgz \
  http://render.bootchart.org:8080/bootchart/render > \
  bootchart.png
  ```

The prior approach does not work with the distribution Ubuntu. Use the following instructions instead:

- get the Bootchart package with an package manager like apt-get or synapsis
- login as root
- edit the `/etc/init.d/stop-bootchart` file to cut off the automatic generation of the chart at boot time, command out the line `create_chart`
- logout as root
- the manual chart generation does work as above

3.1.2.4 Sample of a Bootchart chart

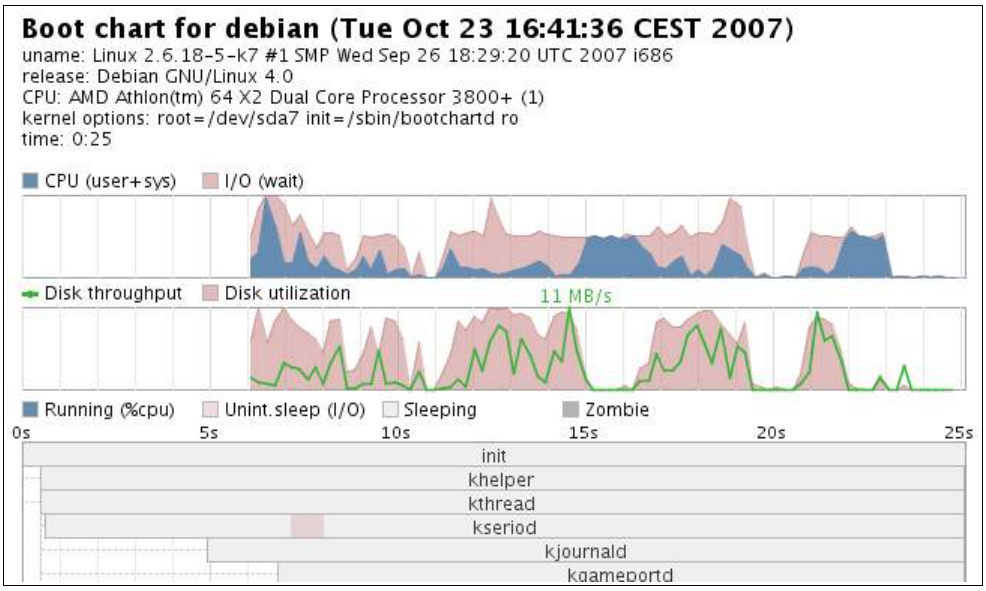


Figure 3.1: Sample of a chart of Bootchart

At the beginning, for the first few seconds, there is no statistic information shown, because the `/proc` file system does not exit at this time.
The first chart shows the activities of the CPUs. The blue graph shows the utilization of the CPU with user and system activities. The red one, shows the waiting for I/O (Input/Output). The chart beneath illustrates the activities of the disk. The red graph shows the utilization of the disk and the green graph the throughput of the disk. The graphs of disk utilization and I/O wait is similar, because at the boot process the most of the I/O traffic is produced by the disk. Based on the relative long access latency of the disk, many I/O wait comes out.
The GANT diagram shows the hierarchies of processes started and their activities, like running, inactive, because of I/O wait and sleeping.

3.1.3 Fbtt (Florian's Boot Time Tool)

3.1.3.1 Short description of Fbtt

This is a tool written by myself during this diploma thesis. It uses the `/proc` file system like Bootchart does, but it reads only ones. In addition it finds out, how many cycles the CPU has needed since the last power on or reset.
This feature was taken from the student research project of Daniel Parthey [PAR06].
Fbtt is executed by a script used by the display manager GDM.

3.1.3.2 Measuring points

The starting point of measuring is equivalent to starting point of the kernel and the end point equals the ready to use state of the display manager GDM.
By calling Fbtt at another point during the Linux boot process, another endpoint is realizable. The `/proc` and the regularly file system must exist at this time.

3.1.3.3 How to install Fbtt

- compile the source code of Fbtt (to find in chapter G.1)
  with executing the Makefile or download the binary from [FBT08]
- login as root
- make a directory named `/usr/local/sbin` with the command

```
mkdir /usr/local/sbin
```

- go into the directory, where the binary of fttb is and copy the binary into the directory /var/log/fbtt with the command

```
cp fbtt /usr/local/sbin/fbtt
```

- add following line at the beginning of the file `/usr/lib/gdmplay` used by GDM

```
/usr/local/sbin/fbtt > /dev/null
```

- logout as root
- after a run with Fbtt, the log file can be found in the directory `/var/log/fbtt`

3.1.3.4 Sample of a Fbtt output file

Output	**Description**
`fbtt version: 0.12`	Version of used Fbtt
`date: 2007-11-06`	Date of generating this output
`time: 16:47:26`	Time of generating this output
`boottime: 21.06`	The elapsed time in seconds since kernel startup
`idletime: 14.54`	The idle rate in seconds of the boot time
`tsc: 73985517000`	State of the Time Stamp Counter since the last power on or reset. This information can only be converted in seconds, when no power safe mechanism is used.

Table 3.1: Sample of a Fbtt output file

3.2 Comparison of measuring methods

3.2.1 Description

A measuring is only as good as the measuring method used. Computer-aided measuring is preferred, because it is more precise than manual. To find out, how trustworthy and precise the measuring methods are, they will be compared to against each other. For that, a test series is arranged. However these are tested in parallel. It is also examinated, what impact the power saving function of the CPU (Cool & Quiet) and Bootchart have on the accuracy of the time measurement.
The starting and end points are described in detail in chapter 3.1. With Bootchart, three modes have been tested.
The first mode is the complete operation mode, in which Bootchart chart generates a png-file automatically after it has stopped the collection. In the second mode, Bootchart does not generate a start point. This has to be made manually. In the last variant Bootchart is completely deactivated.
An additional test is made for analyzing the influence of Fbtt. For that a little script was written.
This script executes three versions of Fbtt. The three versions differ in the filename only, so that the data is always read from the mass storage and not from the system buffer or from the cache of the CPU. The script is run in these variants with and without reboot, to get a result with and without using the system buffer and cache. The power save function Cool & Quiet is deactivated, so that the time stamp counter (TSC) can be read out and converted to seconds.

3.2.2 Results

Cool & Quiet	Bootchart	Bootchart [s]	Fbtt [s]	Stop Watch [s]
with	complete	22 21 22	29.90 28.56 29.30	30.90 29.58 30.37
	without chart generate	21 21 22	27.46 28.13 28.57	28.32 28.40 29.28
	without	- - -	26.48 25.05 27.03	27.21 25.56 27.83
without	complete	20 20 21	24.98 24.93 25.99	26.28 26.08 26.83
	without chart generate	21 21 21	24,20 25.79 23.88	24.96 26.70 25.01
	without	- - -	25.69 24.01 25.45	26.49 24.67 26.14

Table 3.2: Single values of test series with SEA3.5SATA500GB

Cool & Quiet	Bootchart	Bootchart [s]	Fbtt [s]	Stop Watch [s]
with	complete	21.66	29.25	30.28
	without chart generate	21.33	28.05	28.67
	without	-	26.18	26.87
without	complete	20.33	25.30	26.40
	without chart generate	21.00	24.62	25.56
	without	-	25.05	25,77

Table 3.3: Measured values of central tendency of test series with SEA3.5SATA500GB

Without system buffer and CPU cache			With system buffer and CPU cache		
TSC	**Difference**	**Time [s]**	**TSC**	**Difference**	**Time [s]**
297558237068	-	-	532102424075	-	-
297584706409	26469341	0.013	532114105278	11681203	0.006
297611980908	27274499	0.014	532152510749	38405471	0.019

Table 3.4: Influence of Fbtt (Run 1)

Without system buffer and CPU cache			With system buffer and CPU cache		
TSC	**Difference**	**Time [s]**	**TSC**	**Difference**	**Time [s]**
316859766786	-	-	344555883998	-	-
316936353326	76586540	0.038	344565011106	9127108	0.005
316963935110	27581784	0.014	344576459964	11448858	0.006

Table 3.5: Influence of Fbtt (Run 2)

Without system buffer and CPU cache			With system buffer and CPU cache		
TSC	**Difference**	**Time [s]**	**TSC**	**Difference**	**Time [s]**
556644290029	-	-	610602463575	-	-
556666621010	22330981	0.011	610610955037	8491462	0.004
556694452667	27831657	0.014	610622521438	11566401	0.006

Table 3.6: Influence of Fbtt (Run 3)

3.2.3 Discussion

First at all, it can be noted, that the time of the boot process differs up to two seconds, even though the same variant and measuring method is applied. Considering how complex the boot process is and which components (e. g. Scheduler, Native Command Queuing by the hard disk, latency of each memory level and whole optimization in the kernel) are involved, this is not amazing. An impact of the configuration of the network interface can be excluded, because since version 7.04 of Ubuntu this is done by the Network-Manager after the boot process.

The power saving function Cool & Quiet of the AMD Athlon 64 X2 CPU does influence the boot process during regular operation only marginally, i. e. without using Bootchart it needs one second more with the power save function.

The measured values of Bootchart and Fbtt varies enormously. Attention should be paid to the fact that only the values of the measurement with Bootchart and without chart generated can be compared with each other.

Without the power save function, the difference is more than four seconds (approximately 25 percent) and with using the power save function, the difference is more than six seconds (approximately 15 percent). This disadvantage of Bootchart is mentioned

amongst others in [LMA07b]. The reason of this phenomena is, that Bootchart stops the measurement, when e. g. the GDM (GNOME Display Manager) is started and listed in the process list. For accelerating the boot process, GDM and other Display Manager move more and more towards the starting point of the Linux boot process. That implicates, that other processes are running in the background, which also need resources (e. g. CPU, mass storage) like Bootchart does.
Therefore the Display Manager needs more time to reach the ready to use state, than without sharing the resources with other processes, which are running in parallel.
Bootchart is still an essential tool for analyzing the boot process, but is not adequate in measuring the precise boot time.
The comparison between Fbtt and the stop watch measurement always shows a difference of approximately one second. The reason of this is, that the starting point of both measuring methods are not exactly the same. The stop watch measuring method includes additional time to load the initrd image into the RAM, executed by the boot loader. Fbtt takes the startup of the kernel as starting point.
The influence of the several measurement methods are different. With Bootchart, it takes two seconds more by using the power save function Cool & Quiet of the CPU and one other second for generating the chart of Bootchart at boot time. Without using Cool & Quiet a negative influence of Bootchart is not verifiable. The influence of Fbtt is so marginal, that it can only be proofed by reading out the TSC. Fbtt needs less than 0.04 seconds on a running desktop with a couple of other programs in the background. The ratio of the needed 0.04 seconds are less than 0.02 percent of the Linux boot time. The method of using the stop watch has no negative effect on the boot process.

4 Myth of continuously rising boot times

At the same time personal computers get powerful more and more, the amount of transistors for a CPU is doubling approximately every two years [MOO65]. However still a lot of people report, that the boot time of a new Linux distribution on a new computer has increased compared to older system.

Is that assumption right or is it a miss impression?

The first test series tries to unravel the myth. Therefore several distributions (32-bit versions only) of the last two years (see table 4.1) are installed on a hard disk and tested on a dual and single core based computer system.

Distribution	Version	Major release date
Debian	3.1r6a	6. June 2005
Debian	4.0r1	8. April 2007
Ubuntu	6.06 LTS	1. June 2006
Ubuntu	6.10	26. October 2006
Ubuntu	7.04	19. April 2007
Ubuntu	7.10 beta	18. October 2007
SuSE Linux	10.0	6. October 2005
Open Suse	10.1	11. May 2006
Open Suse	10.2	7. December 2006
Open Suse	10.3 RC3	4. October 2007

Table 4.1: Tested distributions for the myth of continuously rising boot times

4.1 Introduction of the different distributions

4.1.1 Debian

Ian Murdock has initiated the Debian project in 1993. Since 1997 the non company owned project is supported by the non-profit organization *Software in the Public Interest (SPI)*. Today, the Debian project is maintained by approximately 1000 persons. [WIKIa][DEBI]
The prepackaging of this distribution is very conservative. It attaches great importance to stability and safety. [DIST07]

4.1.2 Ubuntu

Ubuntu is based on Debian GNU/Linux. But in the back of Ubuntu is a company, named Canonical Limited, established in 2004 by the South African billionaire Mark Richard Shuttleworth. Instead of making profit Mark. R. Shuttleworth tributes several millions of dollar to Canonical Limited every year. [WIKIb]
Ubuntu is focused on an user-friendly installing process and maintainable operation system. It is particularly suitable for Linux beginners.
This distribution has a very modern prepackaging. Two versions do exist, one puts more emphasis on being up to date and innovation and the other on stability and safety. [DIST07]

4.1.3 OpenSUSE / SuSE Linux

OpenSUSE is based on SuSE Linux. The first version was released in 1994. Today, the commercial version is distributed under the name SuSE Linux, and openSUSE the free community version. [WIKIc]
This distribution is not so conservative as Debian and not so modern as Ubuntu. It brings a couple of interesting tools developed by SuSE e. g. Yast [YAST] and AppArmor [APPA].

4.2 Configuration of the distribution

The default settings were taken from each distribution. An exception was made by the choice of the window manager. Gnome was chosen for all system. The partitioning of the hard disk which covers all evaluated Linux distributions can be seen in table 4.2. Always the latest versions are installed locally behind the older ones with the effect that the latest versions suffer from lower bandwidth due to hard disk characteristics. The output of the hard disk benchmark h2benchw [HEI] diagram of the SEA3.5SATA500GB shows (see chapter B.2), that the throughput decreases with ascending block numbers.
The third extended file system (ext3) is always used. In addition the automatic login is deactivated and at last the check of the file system is switched off with setting the fsck option in `/etc/fstab` to zero.

Wholly different is the installation process of Debian 3.1, because the standard installation does not work. Several conflicts could not be detached. Therefore the installation was made manually without any network connection because of problems with the handling of packages with an Internet connection. The selection was made in a way that it fits closely to the desktop installation package set.

Kernel version 2.6 was chosen because all of distributions are based on it. The select the 2.6 kernel for Debian 3.1 the selection was made by typing `linux26` at the boot prompt of the Debian 3.1 installation CD.

Range (GiB)	Distribution	File System
0 - 1	swap	swap
+ 100	free	-
+ 20	Ubuntu 6.06.1	ext3
+ 20	Ubuntu 6.10	ext3
+ 20	Ubuntu 7.04	ext3
+ 20	Ubuntu 7.10 beta	ext3
+ 20	SuSE Linux 10.0	ext3
+ 20	SuSE Linux 10.1	ext3
+ 20	Open SuSE 10.2	ext3
+ 20	Open SuSE 10.3 RC3	ext3
+ 20	Debian 3.1r6a	ext3
+ 20	Debian 4.r1	ext3

Table 4.2: Partition table for the test series: The myth of continuously rising boot times

4.3 Used measuring methods

The measuring is done manually with a stop watch. Starting point and end point are chosen as described in chapter 3.1.1.2.
A single core consecutively named ATH64_1.8 and a dual core system named ATH64X2_2.0 are investigated. The system configurations can be seen in detail in the appendix A. As mass storage the desktop hard disk (3.5” disk) named SEA3.5SATA500GB is used. The technical data of the hard disk can be look up in the appendix chapter B.2.
After the installation of the hard disk into the second system, the configuration of the graphics card had to be modified, because the graphics cards are different in both system (confer appendix chapter A.1 and A.2) and require distinct drivers. After all errors referring the X-Server has been removed the next series of measurement were stared.
The first measurement is done without any update of the operating system and the second run was taken after the update has been applied.

4.4 Results

A value consists of three measurements. Each run is rounded to a second and then the arithmetic mean is calculated, which is once again rounded to a second.

Distribution	**ATH64X2_2.0 [s]**	**ATH64X2_2.0 + updates [s]**	**ATH64_1.8 [s]**	**ATH64_1.8 + updates [s]**
Ubuntu 6.06.1	43	35	33	37
Ubuntu 6.10	26	26	28	27
Ubuntu 7.04	32	32	33	33
Ubuntu 7.10 Beta	27	27	30	28
SuSE Linux 10.0	23	43	45	43
SuSE Linux 10.1	44	35	55	37
Open SuSE 10.2	47	42	79	56
Open SuSE 10.3 RC3	26	25	34	34
Debian 3.1 r6a	42	42	43	43
Debian 4 r1	26	25	28	27

Table 4.3: Boot time of different distributions

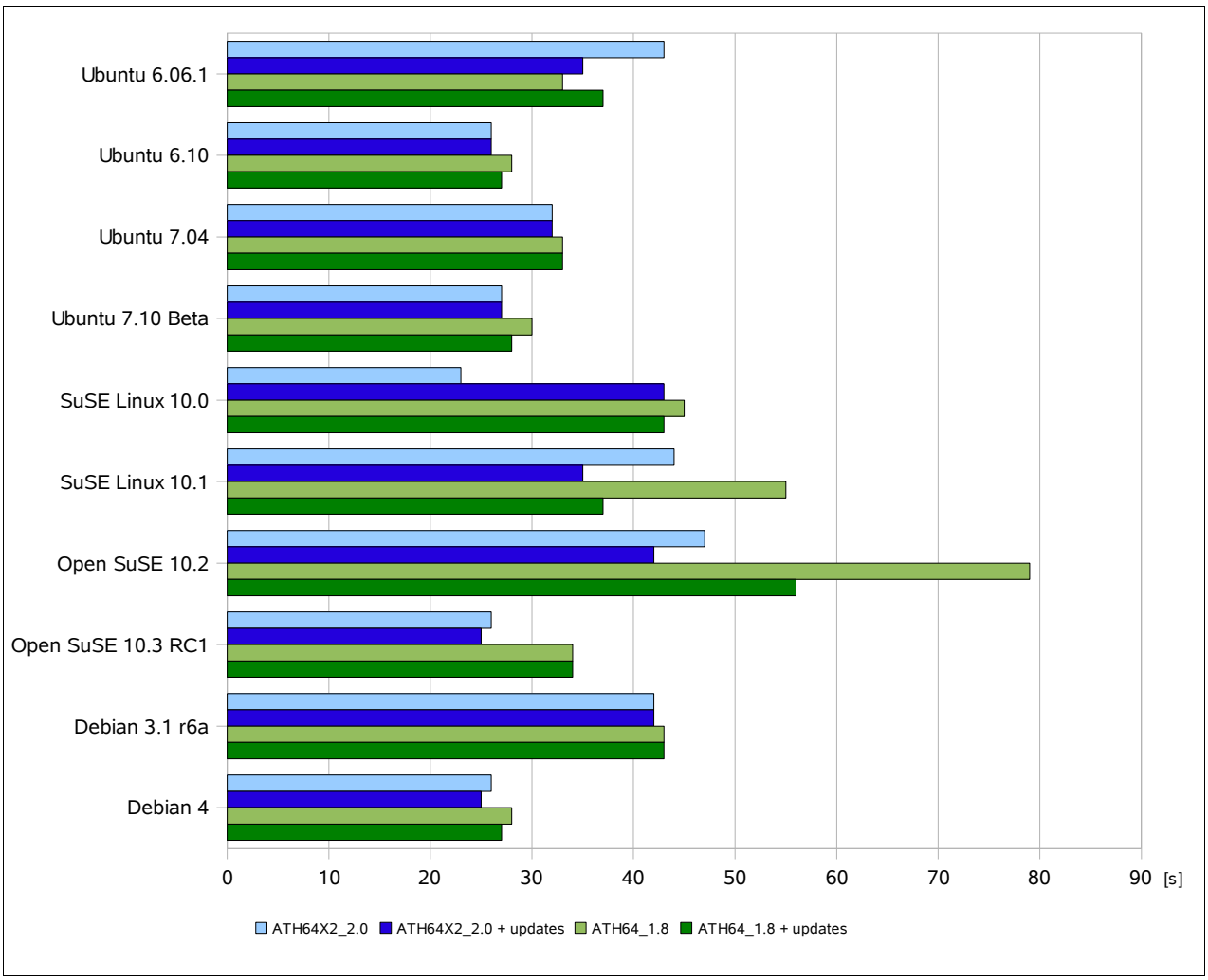


Figure 4.1: Boot time of different distributions

4.5 Discussion

The comparison between the installation of the distributions with and without updates of the operating system shows, that the measurements of the systems without updates have some unusual mavericks with extreme short and extreme long boot time. With the installation of the updates, these mavericks have disappeared. A possible cause for the mavericks are bugs, which are fixed with the updates.

The result of Debian 3.1 is similar to the one in [PAR06a]. The single core system, which was tested in the publication [PAR06] is comparable with the ATH64_1.8 system. Therefore it can be assumed, that the manual selection of the packages correspond to the default installation named Desktop in terms of the boot time behavior.

Following only the installations with updates are considered. It is interesting to follow the history of the several distributions. In Ubuntu 6.10 the System-V-Init is replaced with upstart. Upstart brings a distinct advantage of nine seconds with ATH64X2_2.0 and ten seconds with ATH64_1.8. The version 7.04 loses some seconds and the version 7.10 recovers a couple of seconds. The history of the boot time with the SuSE distributions as

follows: With the change from version 10.0 to 10.1 the time reduces and with version 10.2 increases intensively. With the version 10.2 a new package management system named ZMD is adopted. Mostly ZMD is responsible for the deterioration. With the version 10.3 of the community version openSUSE ZMD is replaced with lipzypp and zypper. With other changes [GIA07], the boot time is shortened to the best result of the SuSE distributions of the last two years. The speed up of the Debian boot process has increased from version 3.1 to version 4 to a high degree. Debian 4 is still using System-V-Init as init system.

The Myth of continuously increasing boot times is busted when we are looking at latest versions. It can be said, that the latest versions need least time for the booting process than all other previous versions of each distribution in the last two years.

But why is the impression emerging, that the start up of a computer is increasing continuously. This is because, the time, which the firmware e. g. PC-BIOS needs to start up, is not included in the tests before. The issue, that the configuration of a PC-BIOS can differ up to 28 seconds, has Daniel Parthey shown in his student research project [PAR06b]. Therefore, it can not be excluded, that one or another new computer does need more time for booting up, as an old one. In the majority of cases, Linux is not the reason for the increase of the time of the boot process. Only with SuSE from version 10.1 to version 10.2 the time of the boot process of Linux increases and insignificantly with Ubuntu from version 6.10 to version 7.04.

5 Research of the latest versions

5.1 Evaluation of boot times with using traditional mass storage

5.1.1 Configuration

The configuration is the same as described in chapter 4.2, with the difference, that only the latest version of each distribution is installed. The partitioning of the disks are shown in table 5.1. For this the mass storages SEA3.5SATA500GB and SAM2.5SATA100GB are used with the System ATHX2_2.0.

Range (GiB)	Distribution	File System
0 - 2	swap	swap
+ 10	free	-
+ 10	Ubuntu 7.10 beta	ext3
+ 10	openSUSE 10.3	ext3
+ 10	Debian 4	ext3

Table 5.1: Partition table of both hard disks for the research of the latest versions

5.1.2 Measuring methods

The measuring is done manually with a stop watch as described in chapter 3.1.1.2.

5.1.3 Results

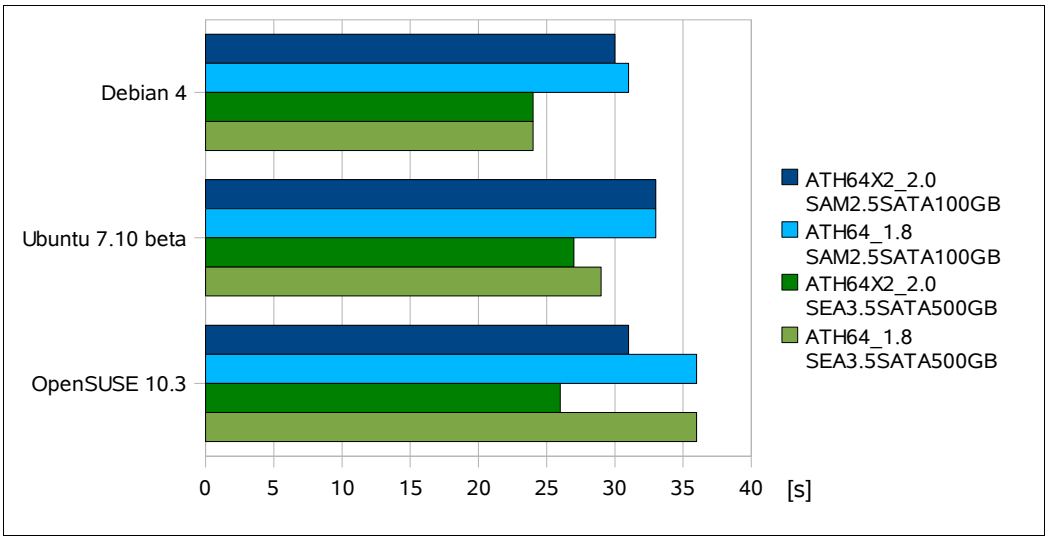


Figure 5.1: boot time of the distributions Debian 4, Ubuntu 7.10 beta, openSUSE

	ATH64X2_2.0 SAM2.5SATA100GB [s]	**ATH64_1.8 SAM2.5SATA100GB [s]**	**ATH64X2_2.0 SEA3.5SATA500GB [s]**	**ATH64_1.8 SEA3.5SATA500GB [s]**
Debian 4	30	31	24	24
Ubuntu 7.10 beta	33	33	27	29
OpenSuse 10.3	31	36	26	36

Table 5.2: Boot time of the distributions Debian 4, Ubuntu 7.10 beta, openSUSE

5.1.4 Discussion

Debian 4 and Ubuntu 7.10 beta show a similar behavior. That is not amazing, because Ubuntu is a descendant of Debian. The single core and dual core need the same boot time. The reason is, that `CONCURRENCY=none` setting is used by default. A minor advantage of the time difference comes from the unequal clock speeds. With openSUSE 10.3 the difference between single and dual core is larger with up to five to ten seconds respectively. OpenSUSE 10.3 has the parallel execution enabled by default. The system ATH64_1.8 with the single core is restricted by the power of the CPU and not by the capability of the mass storage. With Ubuntu 7.10 and Debian 4 the limitation is the mass storage.

The measurement values of each distribution on the ATH64X2_2.0 system with SEA3.5SATA500GB as mass storage are almost equal. This is amazing, because each distribution uses another init system for the boot process. Debian 4 use the over 20 years old System-V-Init, but it has the best boot time of all.

In next chapter the latest versions of the distributions are analyzed on the ATH64X2_2.0 system with SEA3.5SATA500GB as mass storage and with Bootchart in terms of the behavior of the boot process.

5.2 Analysis of the boot process with Bootchart

5.2.1 Configuration

The same configuration as in chapter 5.1.1 described it used but with the exception, that only the ATH64X2_2.0 system with the SEA3.5SATA500GB as mass storage is applied.

5.2.2 Used measuring methods

Bootchart is used for getting more information on the boot process at the boot time.

5.2.3 Results

5.2.3.1 Bootchart of Debian 3.1

The chart of Debian 3.1 is chosen as an example for an older version with no optimization of the boot process. This result corresponds to one which is shown in [PAR06a].

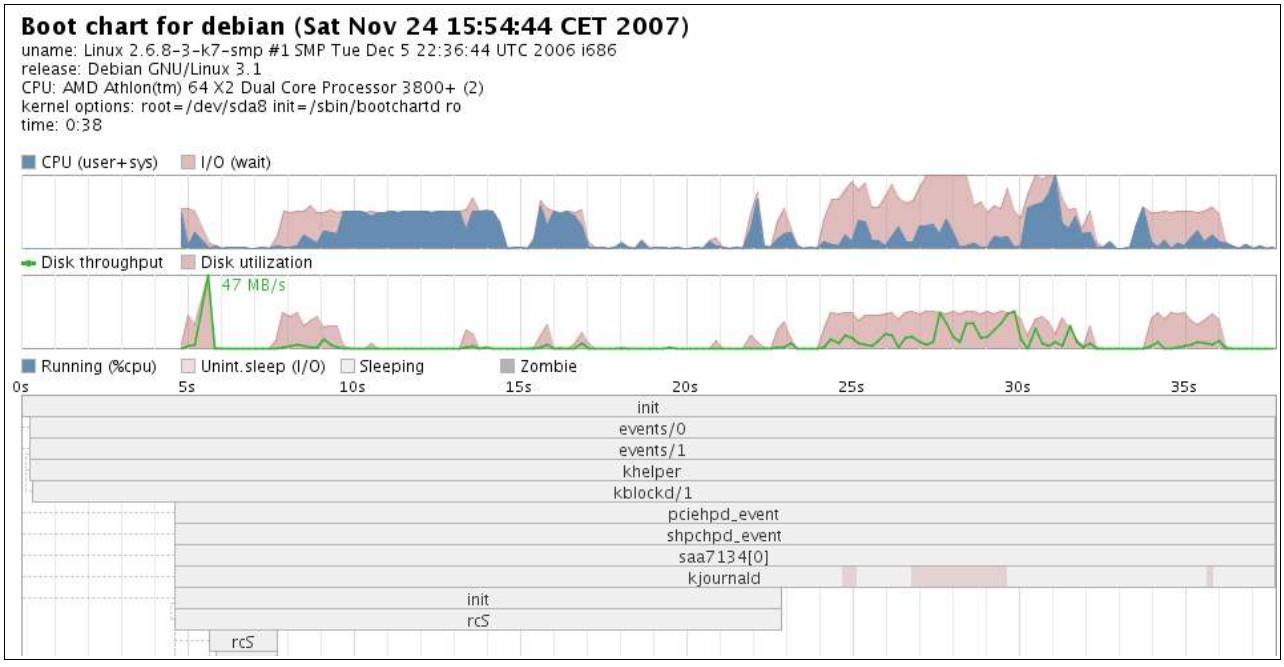


Figure 5.2: Abstract of the boot chart of ATH64X2_2.0 and SEA3.5SATA500GB (Debian 3.1)

5.2.3.2 Bootchart of Debian 4

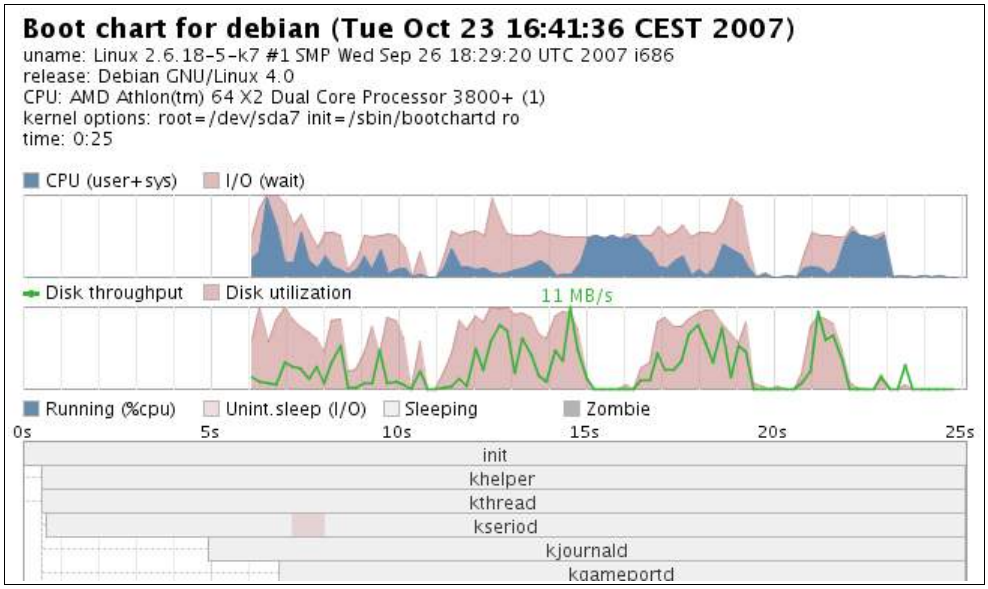


Figure 5.3: Abstract of the boot chart of ATH64X2_2.0 and SEA3.5SATA500GB (Debian 4)

5.2.3.3 Bootchart of Ubuntu 7.10

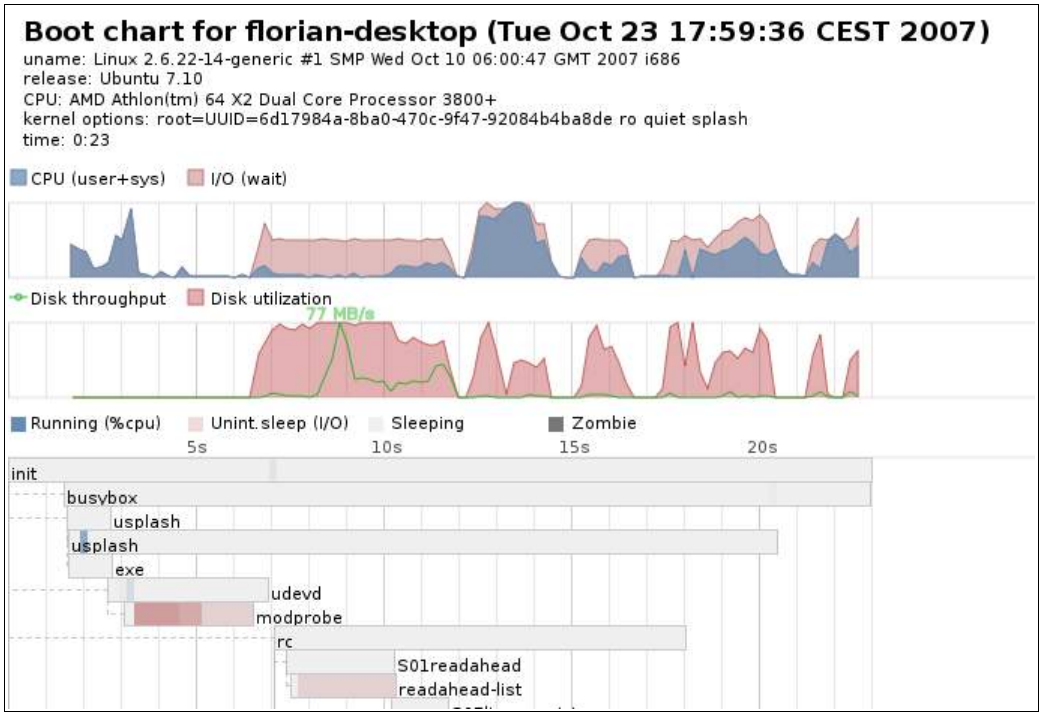


Figure 5.4: Abstract of the boot chart of ATH64X2_2.0 and SEA3.5SATA500GB (Ubuntu 7.10)

5.2.3.4 Bootchart of OpenSuse 10.3

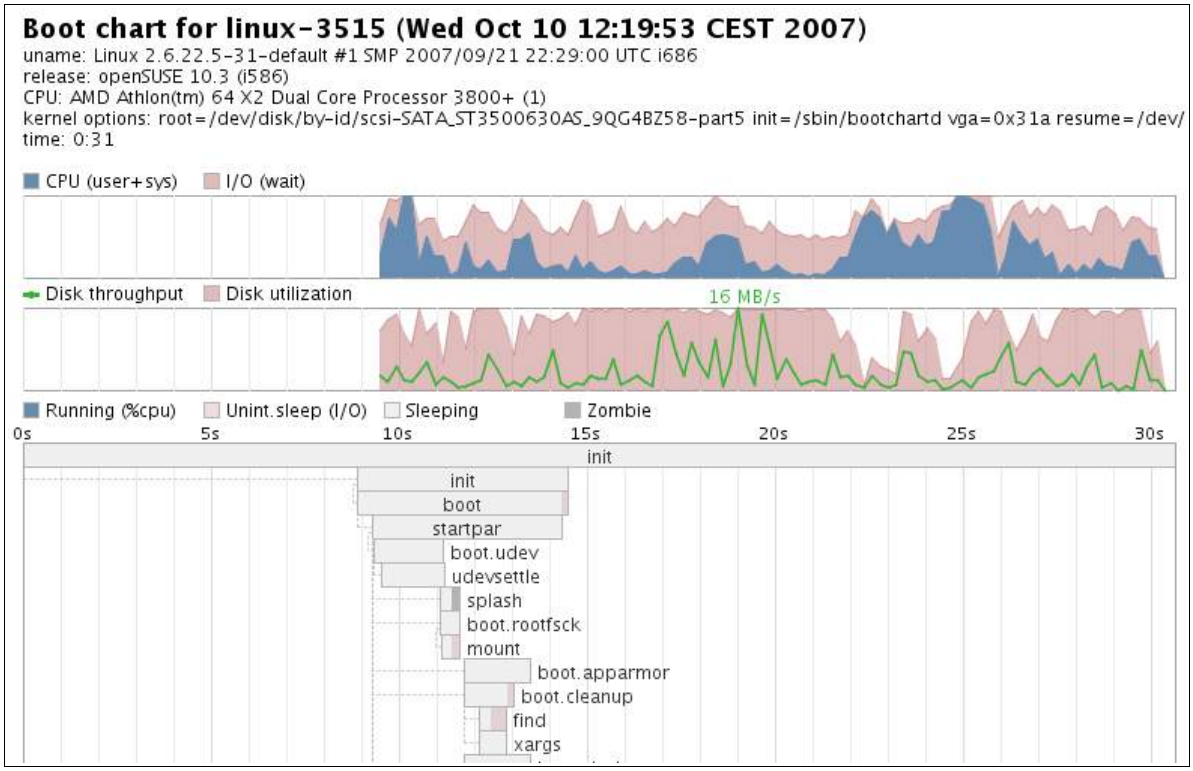


Figure 5.5: Abstract of the boot chart of ATH64X2_2.0 and SEA3.5SATA500GB (openSUSE 10.3)

5.2.4 Discussion

It can be noted, that Bootchart does not get the correct boot time. With Debian 4 (25 versus 24 seconds) and OpenSuSE 10.3 (31 versus 27 seconds) the measurement is finished too late and with Ubuntu (23 versus 29 seconds) too soon. The values of reference are the ones which are got with the stop watch in chapter 5.1.3. The tested versions of the several distributions are using different init systems. Debian 4 is still using System-V-Init, Ubuntu from version 6.10 upstart [UPS07] and OpenSuSE since a long time startpar, a init system like Initng [GIA07].

The boot chart of Debian 3.1 shows a lot of sections, where neither the CPU nor the hard disk have any activity, e. g. between the time spread of 17 to 21 seconds. In this situation a process is waiting for a specific event and holds up the whole boot process. The current distributions have no or only a tiny sections, e. g. Ubuntu 7.10 at the moment 15 seconds after the start of the Linux kernel.

A comparison of the Gant diagrams of the several distributions show, that Debian 4 has the least processes with 82, which are started at boot time. With this amount of processes Debian 4, which is based on the old System-V-Init, achieves the best boot time. OpenSuSE 10.3 executes 130 processes in 27 seconds of boot time until the the display manager GDM is ready for login. Ubuntu starts 88 in 23 seconds, but it is unclear how

many processes are started in the remaining six seconds and no accurate number can be given, because Bootchart stops the analysis to soon. It is assumed, that the total number of started processes is similar to OpenSuSE 10.3.
The distributions have a couple of mechanisms to accelerate the boot process. The kernel has a readahead implementation. With readahead the access to a file is speed up by preloading at least parts of its contents into the page cache ahead of time. A report of this method is shown in [OPD07]. Beyond this another preloading mechanism, file list based, does exist, which preloads all the files needed during the booting process. These are labeled different in each distribution and the implementation is also different. For Fedora and Red Hat it is readahead, for Ubuntu readahead-list und for SuSE preload [KOF07b]. The idea of a file list based preloading is to load the data in advance, which is necessary for the boot process, before or during the boot process. It is to differ between the preloading during the boot process and the preloading done for applications like Firefox, OpenOffice etc. The tested versions use only the preload for the boot process.
The chart of Ubuntu 7.10 shows, that Ubuntu 7.10 uses readahead-list at the beginning of the boot process. In older versions, it was done in the background, parallel to the normal boot process. It turned out, that in the most time, the execution in the background is slower than the execution in the front [JDO06]. OpenSuSE 10.3 uses preload in the background. That is why the utilization of the hard disk is different. At the beginning, Ubuntu has a huge utilization of the hard disk with hight data throughput and then only few hard disk activities. OpenSuSE 10.3 has alternating load of the hard disk all the time.
The usage of readahead-list in Ubuntu 7.10 with the system ATH64X2_2.0 and SEA3.5SATA500GB as mass storage achieves a decrease in time about three seconds only. The corresponding chart of Bootchart without readahead-list can be seen in the appendix in chapter E. The advantage of a readahead list is significantly larger when an old hard disk without command queuing and with larger access time is used compared to new one.

6 Influence of several mass storages to the boot behavior

6.1 Local mass storage

6.1.1 Configuration for the hard disks and flashes

The configuration of the hard disks are the same as described in chapter 5.1.1, but only the installation of Ubuntu 7.10 beta is used.
A SATA-CF-Adapter (see appendix chapter C.1) connects the compact flash to the SATA controller. This adapter makes some trouble with NVIDIA chip sets (confer appendix chapter C.1.1).
The partition of the compact flash storages are shown in table 6.1. All updates are installed, which are available at the date of testing. TRACF4GB (a hight end CF card) and TAKCF4G (a low cost CF card) are used as flash memory. A solid state disk (SSD) was announced to get, but it is not available yet. SEA3.5SATA500GB and SAM2.5SATA100GB are used as desktop and notebook hard disks.

Range (MiB)	Distribution	File System
0 - 3645	Ubuntu 7.10 beta	ext3
+ 220	swap	swap

Table 6.1: Table of partitioning of the flash medias

6.1.2 Measuring methods

The measurement method for most of the tests is Fbtt and exerted as described in chapter 3.1.3. For the evaluation of the theoretical boot time of this system with a mass storage, which is unlimited in the data transfer rate, a chart of Bootchart will be generated. A reprofiling of a readahead-list need to be done before (refer chapter 7.2.2.4), therefore most of the access to the hard disk is done with the readahead-list. In addition, further measurements are done with Fbtt with the setting with and without the use of readahead-list in conjunction with a flash medium. Therefore it can be found out, whether readahead-list results in an advantage by using flash memory compared to a hard disk (refer chapter 5.2.4).

6.1.3 Results

Local Mass Storage	Run 1 [s]	Run 2 [s]	Run 3 [s]	mean [s]
SAM2.5SATA100GB	35.94	31.62	35.43	34.33
SEA3.5SATA500GB	27.99	29.36	25.70	27.68
TRACF4GB	21.52	22.46	20.09	21.36
TAKCF4GB	68.04	48.70	65.45	60.73

Table 6.2: Boot times of several local mass storages on ATH64X2_2.0

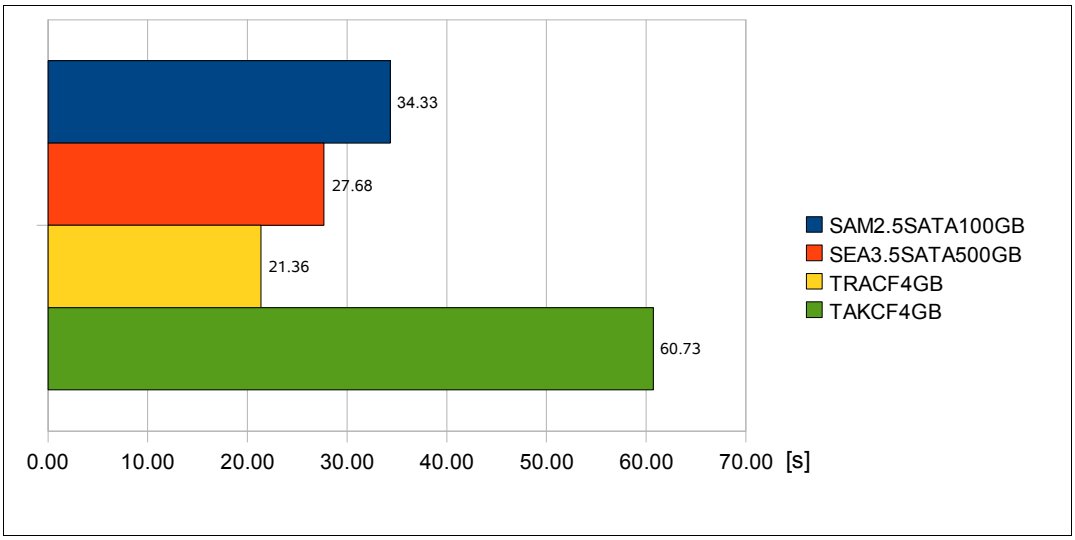


Figure 6.1: Boot times of several local mass storages on ATH64X2_2.0

TRACF4GB	Run 1 [s]	Run 2 [s]	Run 3 [s]	mean [s]
normal, before reprofile	21.52	22.46	20.09	21.36
after reprofile	21.17	21.14	20.04	20.78
without readahead-list	21.31	21.14	20.29	20.81

Table 6.3: Results of the influence of readahead-list

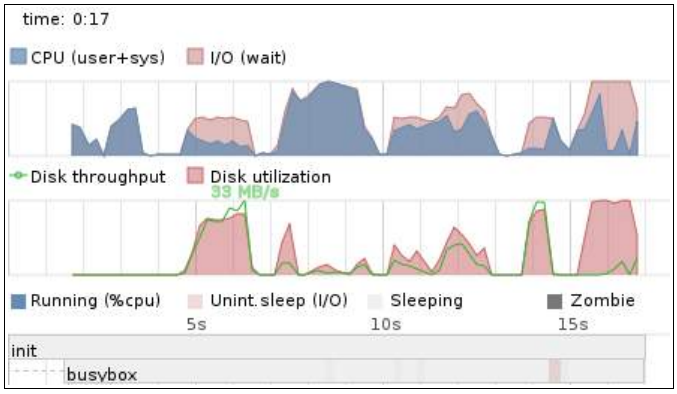


Figure 6.2: Abstract of the boot chart of TRACF4GB without use of readahead-list

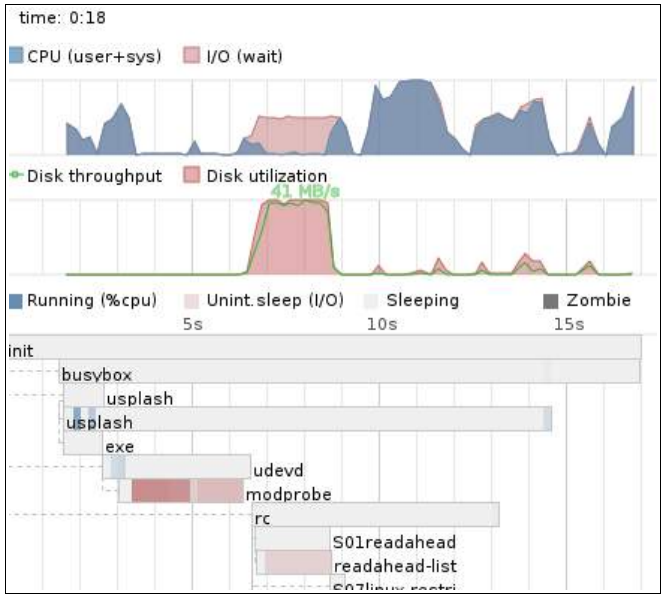


Figure 6.3: Abstract of the boot chart of TRACF4GB after reprofiling readahead-list

The complete charts of Bootchart are in the appendix E.

6.1.4 Discussion

Expectedly the desktop hard disk SEA3.5SATA500GB is faster than the notebook hard disk SAM2.5SATA100GB at the boot process. A huge difference exists between the two compact flash memories TAKCF4GB and TRACF4GB. The TRACF4GB reaches the best time in booting up with 21.36 seconds in the mean. In contrast, the TAKCF4GB is significantly the slowest medium with 60.73 seconds in the mean. A incompatibility with TAKCF4GB, the SATA-CF-Adapter and the NVIDIA chip set (refer appendix chapter C. 1.1) can be excluded, because a similar behavior is shown with a system (P4_3.2) based on a Intel Pentium 4 CPU and Intel chipset. For more details on the system P4_3.2, see appendix A.5. To find out, why the boot process needs so much time with the TAKCF4GB

another boot chart is generated (see appendix E). There are no anomaly to see up to the point, when Bootchart aborts the collecting of data. This corresponds with the output on the screen during the boot process. During the splash screen is shown, the progress bar increases continuously, but by switching to the display manager GDM it slows down enormously. A reason for the very bad result could be the random read and write access to the TAKCF4GB (see appendix B.4 h2benchw value of “installieren”). There is the same phenomenon when using the desktop environment. In contrast, there are no restrictions when using the high end compact flash memory TRACF4GB.
Let us take a theoretical view on the boot time of Ubuntu 7.10. What for a shortest boot time can be reached with a mass storage, which is unlimited in the data transfer rate? After the reprofiling (refer chapter 7.2.2.4) the bulk of the mass storage activity is made by the process readahead-list. With a mass storage, which is unlimited in the data transfer rate, the time of readahead goes to zero. There are no more intensive mass storage activities, so that the time of readahead-list (approximately two seconds) can subtracted from the whole time (approximately 21 seconds) to get the theoretical best time of 19 seconds.
Now, we have a look on the influence of readahead-list in relation to flash memory. Readahead-list has been developed for reducing the seek time of the access to hard disks by avoiding the movements of the read and write head. Flash memory does not have any moving parts, so that readahead-list should not result in an advantage. This presumption is confirmed by the results of the Fbtt measurements. There is no real difference between the time with and without using readahead-list. To verify the measurements, a boot chart of both variants is generated. The different access to the mass storage can be seen very clear.

6.2 Network mass storage

6.2.1 Configuration for the net boot

6.2.1.1 Introduction to configure the client

It is possible to load the data for the boot process from a RAM disk over the network. Therefore a minimal installation of Ubuntu 7.10 is used, which needs < 512 MiB and fits into a RAM disk. A temporary file system (tmpfs) can not be used as file system, because NFS does not support such a file system [NFS].

As starting point the installation selection of a command-line system is used from the installation medium Alternate CD of the Ubuntu 7.10 distribution.

After that, the packages xserver-xorg, x-window-system-core, gdm, nfs-common and feisty-gdm-themes are installed with the following commands. They should be installed one after another, because faults are easier to find.

```
sudo apt-get install xserver-xorg -fix-missing

sudo apt-get install x-window-system-core

sudo apt-get install gdm

sudo apt-get nfs-common

sudo apt-get feisty-gdm-themes
```

After the installation of the previous packages, many temporary files are in the directory `/var/cache/apt/` (approximately 65 MiB). These can be deleted without hesitation.

For playing the sound at the moment the display manager GDM is ready for login, the sound file `question.wav` has to be copied from another system into the directory `/usr/share/sounds/`.

For the automatic measurement Fbtt has to be installed (see chapter 3.1.3).

At the end the previous configuration needs 476 MiB of disk space.

6.2.1.2 Introduction to configure the server

The server uses Ubuntu 7.10 (desktop version) as operating system.

The packages needed additionally can be install with:

```
sudo apt-get install dhcp3-server \
tftpd-hpa syslinux nfs-kernel-server initramfs-tools
```

Some directories must be created and the `pxelinusx.0` file copied to right directory

```
sudo mkdir -p /tftpboot/pxelinux.cfg

sudo mkdir /nfsrootm

sudo mkdir /nfsrootr

sudo cp /usr/lib/syslinux/pxelinux.0 /tftpboot
```

After that, the DHCP-server needs to be configured. The client gets a static IP from the DHCP-server. The IP address of the router and domain-name-server need to be adopted. The used configuration of `/etc/dhacp3/dhcpd.conf` looks like below:

```
#file /etc/dhcp3/dhcpd.conf
```

```
allow booting;
allow bootp;
subnet 192.168.4.0 netmask 255.255.255.0 {
  range 192.168.4.100 192.168.4.200;
  option broadcast-address 192.168.4.255;
  option routers 192.168.4.1;
  option domain-name-servers 192.168.4.1;

  #filename "/pxelinux.0";
  filename "pxelinux.0";

}

# force the client to this ip for pxe.
# This is only necessary assuming you want
# to send different images to different computers.
host pxe_client {
  hardware ethernet 00:11:D8:2A:5C:1E;
  fixed-address 192.168.4.5;
}
```

In [SCL06] and [UBD07] the filename for `pxelinux.0` is denoted with

```
filename "/pxelinux.0";
```

This configuration does not work, the boot process over the network fails. With removing the slash, the boot up works.

Before running tftp, it needs to be configured to run in the daemon mode.
In the `/etc/default/tftp-hpa` file `RUN_DAEMON="no"` has to be changed to `RUN_DAEMON="yes"` and the option line to `OPTIONS="-l -s /tftpboot"`.

The content of `/etc/default/tftp-hpa` is now:

```
#Defaults for tftpd-hpa
RUN_DAEMON="yes"
OPTIONS="-l -s /tftpboot"
```

The tftp-hpa service can be started now.

```
sudo /etc/init.d/tftp-hpa start
```

The config file for the boot loader must be created. The content of the used file in the `/tftpboot/pxelinux.cfg` directory is:

```
LABEL linux
KERNEL vmlinuz-2.6.22-14-generic
APPEND root=/dev/nfs initrd=initrd.img-2.6.22-14-generic.nfs
nfsroot=192.168.4.1:/nfsrootm ip=dhcp rw
```

The filename can be the mac address of the client or common *default*. The file with the name of the mac address will be demanded first and the default at last.

After that, the export file `/etc/exports` must be configured (add the directory, which should be exported) and sync the file with

```
exportfs -rv
```

The used `/etc/exports` has the content

```
# /etc/exports: the access control list
# for filesystems which may be exported
#         to NFS clients.  See exports(5).
/nfsrootr         192.168.4.5(rw,no_root_squash,async)
/nfsrootm        192.168.4.5(rw,no_root_squash,async)
```

It is important, that the async option is used. The asynchronous transfer is faster than the synchronous one.
Now, the tftp server is configured, but the data for the client must be put in the right place.

If the server uses the same kernel as the client, the initramfs image can be generated on the server, otherwise the `initramfs-tools` package must be installed at the client and the initramfs image generated.

Before generating the initramfs image the configuration file `/etc/mkinitramfs/initramfs.conf` must be adjusted.

The line with `BOOT=` must be changed from

```
BOOT=local
```

to

```
BOOT=nf
```

The initramfs can be generated on the server with the command

```
mkinitramfs -o \
/var/lib/tftpboot/initrd.img-2.6.22-14-generic.nfs
```

and copy the kernel to the tftpboot directory with

```
cp /nfsroot/boot/vmlinuz-2.6.22-14-generic \
/var/lib/tftpboot/vmlinuz-2.6.22-14-generic
```

After that, the `initramfs.conf` should be changed back to avoid errors in the future with other configurations.

The data of the client has to be transferred to the server. For that, connect the client with the server, boot up to the display manager GDM and change with `Ctrl+F1` to a tty-terminal. Then type the following commands:

```
sudo mkdir /mnt/nfsroot

sudo mount -tnfs -onolock \
192.168.4.1:/nfsrootm /mnt/nfsroot

sudo cp -ax /. /mnt/nfsroot/.

sudo cp -ax /dev/. /mnt/nfsroot/dev/.
```

The network interface information in the configuration file `/nfsrootm/etc/network/interfaces` on the Server has to be commented out.

```
# auto eth0
```

Otherwise the client tries to initialize and start up the network interface again. The consequence is, that the connection with the server is cut off.

Last but one, the `fstab` file in `/nfsrootm/etc/` must be modified. The following configuration was taken:

```
# /etc/fstab: static file system information.
#
# <file system> <mount point>   <type>  <options>       <dump>
<pass>
proc            /proc           proc    defaults        0       0
/dev/nfs       /                nfs     defaults        1       1
none            /tmp            tmpfs   defaults        0       0
none            /var/run        tmpfs   defaults        0       0
none            /var/lock       tmpfs   defaults        0       0
none            /var/tmp        tmpfs   defaults        0       0
```

Last, in the BIOS of the client, boot from LAN must be activated. If there are options for different LAN boot methods, something named PXE is to choose. On the ASUS A8N-SLI-Deluxe and other main boards with NVIDIA chip set use the NVIDIA Boot Agent. Now the hard disk of the client can be removed.

Tip: In all configuration files the double quotes and single quotes must be the right ones. This is often the source for an error.

Variants

Using the hard disk of the server

Reboot the client and reboot the sever, too. After the first reading, the used data is in the system buffer and with a reboot of the server, the system buffer is clean again.

Using the hard disk and the system buffer of the server

The client is rebooted without a reboot of the server. After the first reading, the data used is in the system buffer and during the next request the data will be read from there.

Using the RAM disk

Append the kernel option for the ramdisk size in the config file `/boot/grub/menu.lst` of the grub boot loader.

```
ramdisk_size = 525000
```

After reboot, the RAM disk can be created with the commands:

```
sudo mke2fs -vm 0 /dev/ram0 515000

sudo mount /dev/ram0 /nfsrootr
```

If the RAM disk really exists, can be verified with the command:

```
sudo mount | grep ram0
```

The output of the previous command must be:

```
/dev/ram0 on /nfsrootr type ext2 (rw)
```

If the RAMDisk is ready for use, the data can be copied to it.

```
sudo cp -ax /nfsrootm/. /nfsrootr/.
```

Last, the config file for the boot loader must be change from

```
APPEND root=/dev/nfs initrd=initrd.img-2.6.22-14-generic.nfs
nfsroot=192.168.4.1:/nfsrootm
```

to

```
APPEND root=/dev/nfs initrd=initrd.img-2.6.22-14-generic.nfs
nfsroot=192.168.4.1:/nfsrootr
```

For each measurement the server has to be rebooted, so that the system buffer is really clean. The previous steps has to be made again. The last step is not to be repeated, because it is non-volatile.

Using the RAM disk and the system buffer of the server

The client is rebooted without a reboot of the server. After the first reading, the used data is in the system buffer and during the next request the data will be read from it.

6.2.2 Measurement methods

The measurement method is Fbtt and exerted as described in chapter 3.1.3.

6.2.3 Results

Three measurements are done and the arithmetic mean is calculated.

NFS	Hard Disk [s]	RAM Disk [s]
without system buffer	33.30s	23.07s
with system buffer	21.52s	21.53s

Table 6.4: Values of NFS boot, Fbtt method

The reference time of the minimal system with SEA3.5SATA500GB is 22.55 seconds, measured with Fbtt and 23,32 seconds using the stop watch.

NFS	Hard Disk [s]	RAM Disk [s]
without system buffer	37.89	31.53
with system buffer	25,88	25.78

Table 6.5: Values of NFS boot, stop watch method

A measurement series with the stop watch results in more time in the range of three up to eight seconds. The message “DHCP../” is used as starting point and the ready to login state of the display manager GDM as end point. The additional time at the beginning includes the time for exchanging the DHCP information up to the point of starting the kernel. By using local storage, the boot time is only one second more.

6.2.4 Discussion

The boot process over a Gigabit Ethernet network does not result in an advantage in time in comparison with the values of the Fbtt measurement. The client reaches the time of a system with SEA3.5SATA500GB as local mass storage only by reading out the data from the system buffer. The overhead of NFS and the latency of the Ethernet is to high. Besides this the time for DHCP and PXE comes along, which is included in the measurement with the stop watch. The reason for the huge fluctuation in the time of booting over network, is the exchange of DHCP information. It can be completed in less than one second or take up to 5 seconds. DHCP can not be parallelized by booting over the network. An alternative can be the use of Etherboot/gPXE instead of PXE or a static IP, when the firmware can support it.

6.3 Spread of the measurement values

6.3.1 Description

The phenomenon of the spread has been discovered by the comparison of the different measurement methods (see chapter 3.2).
With the experiment of tweaking Ubuntu (see chapter 7) by using the flash memory TRACF4GB the sneaking suspicion arises, that the spreading of the measurement values are larger with this type of memory. As a result of this experience, here we go into that matter.

6.3.2 Results

Cool & Quiet	Bootchart	Bootchart [s]	Fbtt [s]	Stop Watch [s]
with	complete	15 17 18	21.07 22.11 23.71	21.28 21.72 23.93
	without chart generate	18 17 18	22.71 22.48 22.55	22.94 22.74 22.85
	without	- - -	21.52 22.46 20.09	21.66 22.96 20.63
without	complete	17 16 18	20.31 19.13 21.06	20.80 19.39 21.44
	without chart generate	17 16 17	23.05 20.31 20.94	22,83 20.52 21.14
	without	- - -	19.50 19.19 20.35	19.69 19.46 20.82

Table 6.6: Single values of test series with TRACF4GB

Cool & Quiet	Bootchart	Bootchart [s]	Fbtt [s]	Stop Watch [s]
with	complete	16.67	22.30	22.31
	without chart generate	17.67	22.58	22.84
	without	-	21.36	21.75
without	complete	17.00	20.16	20.54
	without chart generate	16.67	21.43	25.56
	without	-	19.68	19,99

Table 6.7: Measures of central tendency of test series with TRACF4GB

| max – min |	=	Difference
| 22.46 – 20.09 |	=	2.37
| 23.71 – 21.07 |	=	2.64
| 22.71 – 22.48 |	=	0.23
| 20.35 – 19.19 |	=	1.16
| 20.31 – 19.13 |	=	1.18
| 23.05 – 20.31 |	=	2.74

Table 6.8: Single values of difference of minimal and maximal for TRACF4GB

| max – min |	=	Difference
| 29.90 – 28.56 |	=	1.34
| 28.57 – 27.46 |	=	1.11
| 27.03 – 25.05 |	=	1.98
| 25.99 – 24.93 |	=	1.06
| 25.79 – 23.88 |	=	1.91
| 25.69 – 24.41 |	=	1.28

Table 6.9: Single values of difference of minimal and maximal for SEA3.5SATA500GB

Formula for the mean difference of minimal and maximal values:

$$m_{storage} = (\sum_{i=1}^{x} |max_i - min_i|) / x$$ whereas i is the number of test run from i=1..x

Result for the mean difference of minimal and maximal values for the storage TRACF4GB and SEA3.5SATA500GB:

$$m_{TRACF4GB} = 1.72$$

$$m_{SEA3.5SATA500GB} = 1.45$$

6.3.3 Discussion

The comparison of the values by using TRACF4GB and the values of SEA3.5SATA500GB (see chapter 3.2.2) shows, that the spreading is really larger. For a complete statistical analysis the number of passes are not sufficient. The number of passes has been limited, because the most components were only temporarily available for the test environment.

But the tendency is visible. The anomalies of the flash storages are proved in the diploma thesis *Analyzing Real-Time Behavior of Flash Memories* [PAR07].
Because of the enormous spread it is impossible to draw conclusions from the results of the measurement with the flash memory of the impact of Cool & Quiet and Bootchart.

7 Tweaking Ubuntu 7.10 to reduce boot time

7.1 Description

How can the boot time be reduced, without diminishing the compatibility and usability? Some promising hints are given in [THE07]. In this publication the boot time of Ubuntu 7.04 Ultimate is reduced from 35 to 15 seconds (measurement method was Bootchart).

7.2 Configuration

7.2.1 Point of start

To get the shortest boot time the fastest medium is chosen. So the starting point is the installation of Ubuntu 7.10 beta on the TRACF4GB.

7.2.2 Removing services

7.2.2.1 Services applet

This tool can be found under *System >> Administration >> Services*. The settings are changed as illustrated in table 7.1.

Name of service	Default setting	Change to
anacron	enabled	disabled
atd	enabled	disabled
apport	enabled	disabled
bluetooth	enabled	disabled
ntp	-	-
klogd	enabled	disabled
sysklogd	enabled	disabled
powernowd	enabled	enabled
mysql	-	-
mysql-ndb	-	-
mysql-ndb-mgm	-	-
nfs-kernel-server	-	-
samba	-	-
gdm	enabled	enabled
hdparm	disabled	disabled
lm-sensors	-	-
hotkey-setup	enabled	disabled
avahi-daemon	enabled	disabled
nessusd	-	-
acpid	enabled	enabled
apmd	enabled	disabled
cupsys	enabled	enabled
hplip	-	-
ssh	-	-
portmap	-	-
dbus	enabled	enabled
screen	-	-
apache2	-	-

Table 7.1: Settings of services

7.2.2.2 Boot-up-Manager (B.U.M.)

This tool has to be installed before. The following command is used in a terminal.

```
sudo apt-get install -y  --force-yes bum
```

Now this tool can be found under *System >> Administration >> Boot-up Manager*. The settings are changed as illustrated in the table 7.2.

Name of service	Default setting	Change to
cron	enabled	disabled
laptop-mode	enabled	disabled
rsync	enabled	enabled
snort	enabled	enabled
upstart	enabled	disabled

Table 7.2: Settings of services changed with Boot-up-Manager (B.U.M.)

7.2.2.3 Sysv-rc-conf

This tool has to be installed before. The following command is applied for this.

```
sudo apt-get install -y  --force-yes sysv-rc-conf
```

With this tool only Britty and pcmcia-uti$ have been deactivated.

7.2.2.4 Reprofile readahead-list

After all, the readahead-list has to be updated, otherwise there is no effect, because the files of the services will be loaded in the file system cache again. For that, the profiling mode of the readahead-list has to be executed. In the boot loader the option profile has to be added in the following way:

At the boot up menu (GRUB) select the default kernel. Then press `e` for edit and choose the first line, that starts with `kernel` and press `e` again. After that move to the end of the line, add the word `profile` and press enter. At last, press `b` to boot. [JDO06]

7.2.3 Acceleration the file system

As file system ext3 is used. For ext3 three journaling methods exist, Journal Data Writeback, Journal Data Ordered and Journal Data. Journal Data Ordered is chosen by default. In the ordered mode, ext3 only journals meta data, but it logically groups meta data and data blocks in a single transaction. The associated data blocks are written first, when it is necessary to write the new meta data out to disk. In general, Ordered method of the ext3 file systems is slightly slower than the Writeback, but is significantly faster than the full data journaling. [THE07]

The method is changed from Ordered to Writeback. Unlike described in [THE07] the command

```
sudo update grub
```

is not used, because is rewrites splash into the `/boot/grub/menu.lst`. No changes are needed in this file in the version 7.10 of Ubuntu.

Open with

```
sudo gedit /etc/fstab
```

the `/etc/fstab` in an editor and look for `defaults,errors=remount-ro` and change it to `defaults,errors=remount-ro,data=writeback,noatime` .

At least, the command

```
sudo tune2fs -o journal_data_writeback /dev/yourdrive
```

has to be executed. `Yourdrive` has to replaced with the real drive identifier.

7.2.4 Parallel execution

The file `/etc/init.d/rc` is to edit. The line

```
CONCURRENCY=none
```

is changed to

```
CONCURRENCY=shell
```

This change generates an error after the login. That is maybe the reason, why the default for concurrency is none. The bug is known and described in [BUG]. The cause of the bug is, that the HAL daemon does not wait enough for Dbus.
An easy workaround exists. The starting point of HAL has to be changed to the next level, from S12 to S13. For this the following command needs to be executed only:

```
sudo mv /etc/rc2.d/S12hal /etc/rc2.d/S13hal
```

With this change Dbus is always ready before HAL is executed.

7.3 Results

Run	Run 1 [s]	Run 2 [s]	Run 3 [s]	Run 4 [s]
Starting Point	21.53	20.06	19.91	22.15
Deactivate Services	20.64	20.63	20.06	19.97
FS Journaling to Writeback	20.20	20.51	18.93	20.14
Concurrency=shell	17.42	19.17	19.17	18.76

Table 7.3: Times of the steps of tweaking Ubuntu 7.10 beta

	minimal [s]	maximal [s]	mean [s]
Starting Point	19.91	22.15	20.91
Deactivate Services	19.97	20.64	20.33
FS Journaling to Writeback	18.93	20.51	19.95
Concurrency=shell	17.42	19.17	18.63

Table 7.4: Minimal, maximal and mean times for the steps of tweaking Ubuntu 7.10 beta

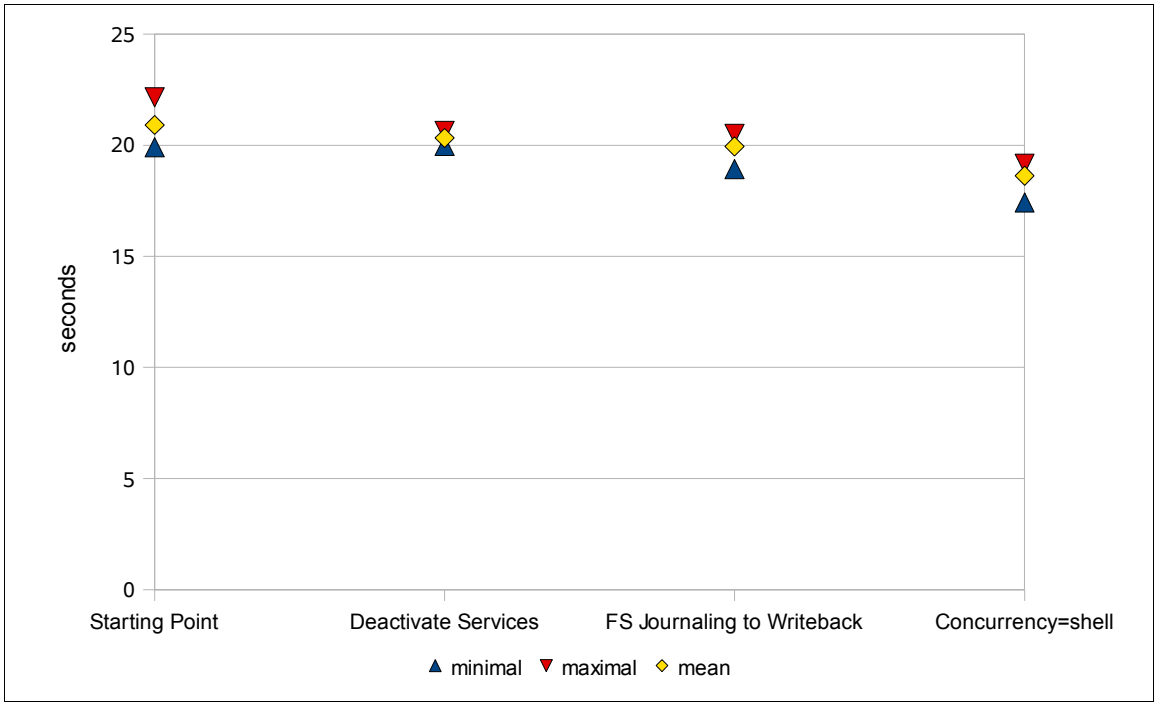


Figure 7.1: Times of the steps of tweaking Ubuntu 7.10 beta

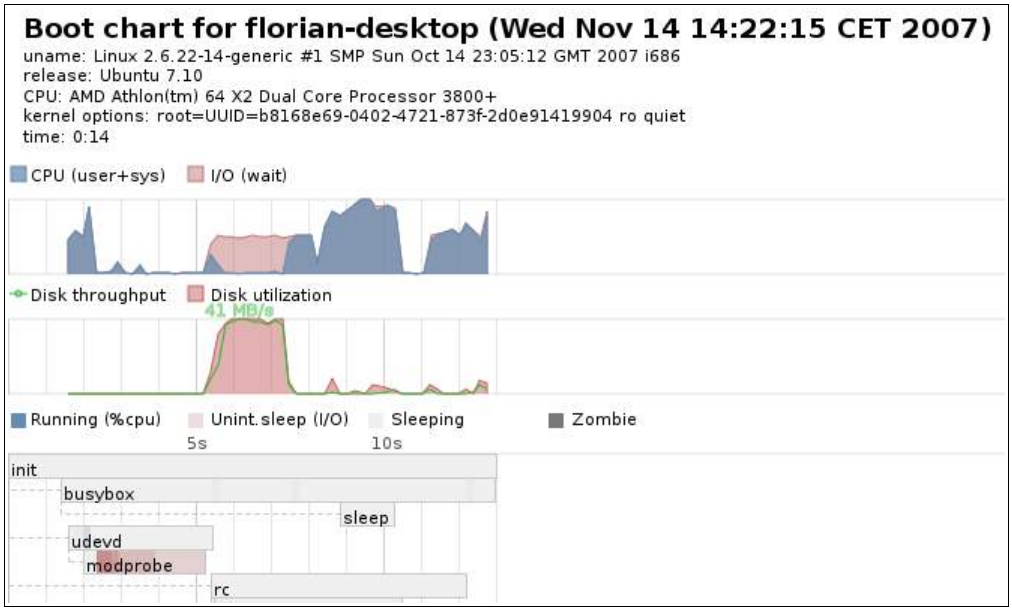


Figure 7.2: Abstract of the boot chart of the tweaked Ubuntu 7.10 beta with TRACF4GB

The complete charts of Bootchart are in the appendix E.

7.4 Discussion

The result is disillusioning, the time can be reduced to 19 seconds only plus the boot time of the firmware. It is still to much for a consumer device, which should be used for a short time several times a day. The effort results in three seconds time reduction only. The last step, with the least effort to enable the parallel execution makes 50 percent of the time advantage, which is one and a half second. If it works fine, it can be used without hesitation. The remove of not needed services makes less than a second and the change of the journaling method, too.
But why could the boot time of Ubuntu 7.04 Ultimate be reduced that extremely, described in [THE07]? That is easy to explain. Ubuntu Ultimate installs a lot of additional software by default. Also there are heavy weights like apache, mysql, samba and others. Deactivating this services brings a lot of time. Besides Ubuntu 7.10 is more optimized than the version 7.04 (see boot times in chapter 4.4), therefore there is less potential for optimization.

Let us take a look at the boot chart of the tweaked Ubuntu 7.10 boot process (see boot chart above). At the beginning it takes a long time, about 3 seconds, to load the driver modules with modprobe. Maybe, some driver modules are not necessary for the system and can be removed. More than one second of time speedup is not achievable. Alternatively there exists an experimental implementation of loading kernel modules in parallel for a CE device [WOO07]. This brings noticeable an advantage. Sometimes, it can need more time, because the management is more expensive. The big disadvantage is, that all implemented drivers up to now, has to be adjusted, so that an implementation for

most systems is a distant prospect. The next optimization step could be to execute hwclock in parallel. This works fine in openSUSE 10.3, but it is not trivial, because the executed software at boot time can come into trouble during the init phase, when the clock does change. The advantage of time improvement is about a half seconds.
When we add all optimizations discussed together and subtracts them from the measured time, a boot time of about ten seconds plus the time for boot up the firmware is far away. Another way is to reduce the amount of services extremely and use small-sized libraries like uclib, but that would reduce the usability of the system and the compatibility to other software.
In the next chapter, a solution is examined, which does not follow the standard boot process scheme.

During these measurements presumption comes, that the spread of the values measured is bigger when using the flash memory TRACF4GB as mass storage as using a hard disk like SEA3.5SATA500GB. Thereupon another experiment is done (see chapter 6.3).

8 Alternative to the boot process

8.1 Boot process abstract

Let us take a close look inside the boot process. After the computer is powered on, the CPU and its register are getting to an well-defined state. At the end of the boot process the computer is ready to use. Daemons and services are started and the user can log in, programs can be loaded and used respectively.

The general user is not interested in what is done between power on and log in, as long as it is done fast and accurate.

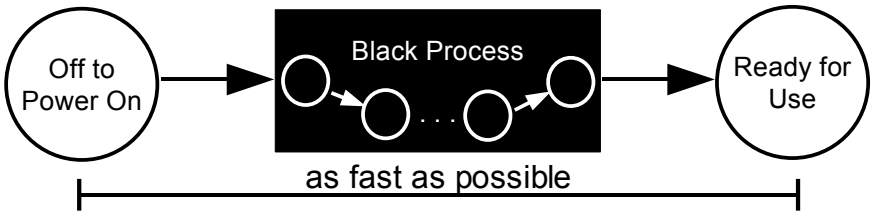


Figure 8.1: Illustration of an abstract boot process

Where is the problem with the current computers? Many parts of the currents computer are volatile, i. e. after power off the prior state gets lost.

How can a personal computer get into the target state? The possibilities are listed in table 8.1.

Possibilities	**Description**
Power on and boot up	General boot up (confer chapter 2.1)
Power on and restore state	After powered on, initialize the CPU, RAM, controller of the non-volatile memory and load an image of a ready to use state from a non-volatile memory into the main memory like Suspend-To-Disk.

Table 8.1: Possibilities to get to a Ready-To-Use-State

But why is Suspend-to-RAM not listed? Suspend-To-RAM is not listed, because it does not pass the initial condition. The computer is not really powered off, the RAM still draws power.

Since the 1990s non-volatile Magneto resistive Random Access Memory (MRAM) is available for main memory. NEC presented MRAM in the year 2007 with an operation speed of 250 Mhz and it is compatible with SRAM. [NEC07]
Besides MRAM, there are a lot of other technologies of non-volatile main memory under way, e. g. SPRAM (Spin transfer torque RAM), FeRAM (Ferroelectric RAM), PRAM (Phase-change RAM), SONOS (Semiconductor-Oxide-Nitride-Oxide-Semiconductor), RRAM (Resistive RAM) and NRAM (Nano-RAM). If one of this technologies is ready for mainstream, Suspend-To-RAM would be another alternative solution.
Today, Suspend-to-RAM does not work on each computer system. These problems are both on Windows and Linux operating systems.

8.2 Suspend-to-Disk versus Power on and restore state

Suspend-to-Disk is nearly that, what we need, i. e. to read an image of a ready to use state. What is Suspend-to-Disk exactly?
After the computer is booted up, the user has the possibility to write an image of the current state to the mass storage and turn it off. After the next power on and loading the kernel the image will be loaded from the mass storage and the last ready to use state will be recovered instead of a normal boot up. After that, the user can continue the work at the point where the Suspend-to-Disk was initiated off.
What is the different between Suspend-to-Disk and power on followed by the restore state?
Instead that the user has to initiate the process of making an image over and over again, an image of the ready to use state of the graphical login will be made automatically. This image is loaded by the next boot up and the state of the graphical login will be recovered. When the image is not generated every time the system can get into an inconsistent state, e. g. a new update for a program is installed, which is included in the image. For such a case the image has to be remade. This process must be organized. Currently no administrative structure exists.
But by now, it can be analyzed, if there is any advantage of time compared with the default boot process. For this analysis the functions of Suspend-to-Disk are enough.

8.3 Suspend-to-Disk and Linux

By now three implementations of Suspend-to-Disk for Linux exists. The beginning of Suspend-to-Disk and Linux was the project swsusp from Mr. Rafael Wysocki.
Because the interests of the developers differed extremely two new project have emerged. For the future it is intended that the distinct projects will come together.
The table F.1 in the appendix chapter F.1 shows the features of the several implementations of Suspend-to-Disk.

The choice of the used variant of Suspend-to-Disk goes to TuxOnIce (in the past named Suspend2). It has a couple of advantages compared to the others. The most substantial argument for TuxOnIce is, that it works independent of an implementation of ACPI in the firmware like a PC-BIOS. All necessary functions exist in the implementation of TuxOnIce. This is very important because in the majority of cases the ACPI tables do not work correctly with Linux. The next positive point is, that a reduced image can be stored, when the capacity of the mass storage is not enough for a full image. Besides there is the possibility to compress the image. This can reduce the write and read times. Also the image can be stored on a swap partition or in a file.

8.3.1 TuxOnIce

8.3.1.1 Installation of TuxOnIce

The operating system Ubuntu 7.10 is used on the system ATH64X2_2.0 and SEA3.5SATA500GB as mass storage.

Patch and compile

After the default installation of Ubuntu 7.10, the packages of the linux-source, kernel-package and build-essential have to be installed.

```
sudo apt-get install linux-source kernel-package \
build-essential
```

The kernel patch for TuxOnIce can be downloaded from [TOIa]. If there is a special patch for Ubuntu, it can be used, otherwise the patch has to be adjusted (see below in section *Adjust the TuxOnIce patch*).

Copy the patch in the source directory `/usr/src/` and try if the patch works.

```
bunzip2 -c /usr/src/tuxonice_rc3.0.patch.bz2 \
 | sudo patch -p1 --dry-run
```

When no error is reported the patch can be used.

```
bunzip2 -c /usr/src/tuxonice_rc3.0.patch.bz2 | sudo patch -p1
```

Then copy the config file of the old configuration to the kernel source.
Change into the directory of the kernel source and execute the following command:

```
sudo cp /boot/config-<kernelversion> .config
```

With Ubuntu there is one specialty. In the default config file for the kernel, `CONFIG_DEBUG_INFO` is enabled. For disable this, change `CONFIG_DEBUG_INFO=y` to `CONFIG_DEBUG_INFO=n`.

Without this change the initrd would increas enormously from 7 MiB to 48 MiB.

Now, the kernel can be compiled and the packages created. Maybe some settings has to be adjusted. Use the submitted values for this.

```
sudo make-kpkg --initrd --append-to-version -with_toi \
--revision 2.6.22-14.46 binary
```

After a couple of minutes up to one hour or more the kernel packages are created.

Install the package, which begins with linux-image. Over the graphical user interface one double click is enough to open the package with the program gdebi-gtk.

After the installation the new kernel is in the `/boot` directory and the grub boot loader is updated automatically.

By now, the necessary scripts for the hibernation of TuxOnIce are missing. These can be downloaded from [TOIa]. After unpacking the downloaded file, the scripts can be installed with executing the `install.sh` with root rights.

Adjust the TuxOnIce patch

First, change into the directory of the source code and execute a dry run of the original patch.

```
cd /usr/src/linux-source-2.6.22

bunzip2 -c /usr/src/tuxonice_rc3.0.patch.bz2 \
| sudo patch -p1 --dry-run
```

If there is an error the patch has to be adjusted.

Copy the source to another directory like `linux-source-2.6.22_copy`, change into it and execute the patch:

```
bunzip2 -c /usr/src/tuxonice_rc3.0.patch.bz2 | sudo patch -p1
```

After that, all parts with an error output are not patched and must be adjusted manually. The reason for most of the errors is, that the patch can not be applied because some of the lines have moved.

Having patched the critical areas manually, a new patch file can be generated. Therefore use the following command:

```
sudo diff-u -r -N linux-source-2.6.22/ \
linux-source-2.6.22_copy/ > my_tuxonice_rc3.0.patch
```

Then compress the patch with bzip2:

```
sudo bzip2 -k -c my_tuxonice_rc3.0.patch
```

At the end, it should be checked whether the new patch works.

Change to the directory of the linux-source with

```
cd /usr/src/linux-source-2.6.22
```

execute and try to run with the new patch:

```
bunzip2 -c /usr/src/my_tuxonice_rc3.0.patch.bz2 \
| sudo patch -p1 --dry-run
```

If there is no error any more the new patch can be used.

```
bunzip2 -c /usr/src/my_tuxonice_rc3.0.patch.bz2 \
| sudo patch -p1
```

Using a swap partition for the hibernation image

For the following command line instructions it is necessary to have root rights. Therefore use e. g.

```
sudo -i
```

Tell TuxOnIce where the swap partition is to find with the command:

```
echo swap:/dev/<swappart> > /sys/power/tuxonice/resume
```

Insert for `<swappart>` the name of the used swap partition.

Then put this information in the file of the boot loader also. For GRUB it is the file `/boot/grub/menu.lst` . After the modification it looks like:

```
title          Ubuntu 7.10, kernel 2.6.22.14-with-toi
root      (hd0,0)
kernel         /boot/vmlinuz-2.6.22.14-with-toi
root=UUID=b317b889-b8e3-4aa6-95a6-c1dcd38e1387
resume=swap:/dev/hda2 ro quiet splash
initrd         /boot/initrd.img-2.6.22.14-with-toi
```

Using a file for the hibernation image

For the following command line instructions it is necessary to have root rights. Therefore use e. g.

```
sudo -i
```

Then create a file with the content `TuxOnIce`.

```
echo "TuxOnIce" > /hibernation-file
```

Thereafter append the file with zeros to the needed size (e. g. 500 MiB)

```
dd if=/dev/zero bs=1M count=500 >> /hibernation-file
```

Tell TuxOnIce which file it has to use for the image:

```
echo /hibernation-file >/sys/power/tuxonice/file/target
```

Check the state of TuxOnIce with:

```
dmesg | tail -1
```

The output should be

```
TuxOnIce: Hibernating enabled.
```

Now get the parameters for the kernel option `resume` with:

```
cat /sys/power/tuxonice/resume
```

The output should be like:

```
file:/dev/hdc1:0xf8020
```

Then put the output of the last cat in the boot command line in the file of the boot loader. For GRUB it is the file `/boot/grub/menu.lst`. After the modification it looks like:

```
title          Ubuntu 7.10, kernel 2.6.22.14-with-toi
root      (hd0,0)
kernel         /boot/vmlinuz-2.6.22.14-with-toi
root=UUID=b317b889-b8e3-4aa6-95a6-c1dcd38e1387
resume=file:/dev/hdc1:0xf8020 ro quiet splash
initrd         /boot/initrd.img-2.6.22.14-with-toi
quiet
```

Apply the hibernate script `/etc/hibernate/suspend2.conf` with adding the line

```
procsetting file/target /hibernation-file
```

and uncomment the `FilewriterLocation` line.

At last the initramfs has to be modified (instructions for Ubuntu 7.10). For this move to the directory `/root` with the command

```
cd /root
```

make a new directory named `tmp_tuxonice` and change into it:

```
mkdir tmp_tuxonice
cd tmp_tuxonice
```

After that the original initramfs file of the kernel with TuxOnIce can be unpacked into the new directory with:

```
cat /boot/initrd.img-<your-version> | gzip -d | cpio -i
```

Then modify the file `/skripts/local`. Add the line

```
echo 1 > /sys/power/tuxonice/do_resume
```

before the root file system is mounted.

Generate a initramfs file of the modified data with

```
find | cpio -H newc -o | gzip > ../initrd.img-<your-version>
```

and change the name of the old initramfs file:

```
mv /boot/initrd.img-<your-version> \
/boot/initrd.img-<your-version>.old
```

At last, copy the new initramfs file into the /boot directory and restart the system.

```
cp ../initrd.img-<your-version> \
/boot/initrd.img-<your-version>
```

8.3.1.2 Configuration and Measurement methods

TuxOneIce version 3.0rc3 is installed as described in chapter 8.3.1.1 and the partitioning of the hard disk can be seen in table 8.2. A file is used as storage for the image. This file is placed near to the beginning of the hard disk for getting the maximal throughput of the hard disk.

For the measurement a stop watch is used because all the other methods do not work by resuming a state from an image. The starting point is when typing the return key in the boot loader Grub, that starts the Linux boot process. When the state of the loaded image is shown, the end point is reached. To get the state of the memory usage, the tool `free` is used and to get more information about the hibernate process the `/sys/power/tuxonice/debug_info` is to read out. For dropping the caches the command

```
echo 3 > /proc/sys/vm/drop_caches
```

is used. The caches can be also dropped by TuxOnIce with setting the option `ImageSizeLimit` in the config file of TuxOnIce to `nocache`, but then there is no chance to get the memory usage with the tool `free`.

Range (GiB)	Description of data	File system	Partition
0 - 24	Ubuntu 7.10 and TuxOnIce file	ext3	/dev/sda1
+3	swap	swap	/dev/sda2

Table 8.2: Partition table of SEA3.5SATA500GB for TuxOnIce

8.3.1.3 Results

	Desktop environment without dropped caches	Desktop environment with dropped caches	GDM without dropped caches	GDM with dropped caches
Used memory [MiB]	304.34	202.79	134.89	62.84
Free memory [MiB]	706.71	808.42	876.32	948.36
Buffers [MiB]	13.25	13.09	5.74	0.21
Cached [MiB]	162.23	50.74	81.38	16.27
Image size uncompressed [MiB]	289.72	166.77	114.78	46.94
Image size compressed [MiB]	150.39	78.54	62.89	21.17
Compression in percent	53	52	45	54
Write speed [MiB/s]	28	30	6	25
Read speed [MiB/s]	40	41	33	32
Time for read image [s]	7.24	4.07	3.48	1.47
Time until TuxOnIce starts [s]	6.06	6.06	6.06	6.06
Time [s]	18.04	15.00	14.91	12.52

Table 8.3: Values of the TuxOnIce boot on ATH64X2_2.0

A description of the Table “TuxOnIce boot” is in the appendix chapter F.2.

8.3.1.4 Discussion

The results of the measurements do not completely satisfy. Especially the result of loading an image with a desktop environment is disappointing, which needs all in all 18 seconds for reconstructing the state of a system with 304 MiB main memory usage. The reason of the bad result is the low I/O throughput. The problem is known and in the version 3.0-rc6 of TuxOnIce the code of block_io has got a rework. The unstable version of TuxOnIce 3.0-rc6 (date 27 February 2008) available from [GIT] does not run on the test system so that it could not be tested. The principle author of TuxOneIce Nigel Cunningham tells, that the I/O throughput should be close to the output of `hdparm -t`. For the used hard disk SEA3.5SATA500GB on the test system the output of `hdparm -t` is 77 MiB per second. This is approximately the value of the *h2benchw* at the beginning of the benchmark. The all in all time of the last example would reduce with the full I/O throughput from 18 seconds to 12 seconds.
At the moment, six seconds of the kernel uptime passes until TuxOnIce starts. This can be improved by modifying the scripts of the initramfs, so that TuxOnIce is started by a short separated branch, which only gets the controller of the hard disk in work, when an image exists. Therewith the time, which passes until TuxOnIce starts, should be shrink from 7 to 2 seconds (included the time of one second for loading the initramfs image into the memory by the boot loader).
The principle author of *swsusp* and *uswsusp* Rafael Wysocki currently works on a implementation, that should allow to hibernate one kernel and resume from another. Nigel Cunningham has figured out a way to transfer information from one kernel to another, but at the moment, there exists no time frame for the complete implementation of this feature. [TOIc]
With such a feature a tiny kernel could be integrated in the LinuxBios or (U)EFI and resume an hibernation image of the default kernel of any Linux distribution. Therefore the time which passes until TuxOnIce starts could be reduced to some milliseconds.

Although the possible I/O throughput of reading the hibernate image from hard disk is not reached and no of the other optimizations are implemented, the loading of an images of a ready to use state of the display manager GDM without dropping the caches needed approximately 15 seconds. It is 12 seconds faster (more than 40 percent) than the normal boot process which needed 28 seconds (confer chapter 6.1.3). A comparison between reaching the state of the desktop environment with an hibernation image and with the normal boot process, the advantage of loading an image would be even more larger. By dropping the caches before writing the image, the time to get the state of the desktop environment back is approximately the same as to resume from the state of the GDM

without dropping the caches. The caches can be deleted unrestrictedly before writing a state of the desktop environment, because it influences the work of the user only marginally. When the state of the display manager should be saved to an image, the caches should not be dropped, otherwise the loading of the desktop environment after the login does need noticeably more time.
By now, the question comes up, would it makes sense to develop a management system, which controls, that only an old image can be loaded when there are no changes on the data, which is saved to the image (confer chapter 8.2). Currently TuxOnIce supports only a kiosk mode [TOId]. In this mode, the same image will be loaded and the file system is mounted as read only, so that it never gets into an inconsistent state. With write access rights to the mass storage the management would be very difficult to guarantee always a consistent state. When the speed of writing an image to the mass storage is close to the value of `hdparm -t`, the image can be written to the mass storage in less than five seconds, so that there is no disadvantage compared to shutdown a system on the traditional way. An additional time to shutdown the system is not needed.

9 Evaluation of embedded systems (IA32)

In this chapter, the normal boot process and the alternative solution to load an image of a ready to use state is evaluated on embedded systems with IA32 architecture.
Intel intents to release the new platform Menlow for Ultra Mobile Personal Computer (UMPC) on the market still this year (2008) [TEC08]. Such a system will have at least the computing power of the system ATH64_1.8. The platform Menlow will be used in many embedded systems in all probability, because the used CPU Silverthorne needs only 2 watt power consumption. The first embedded systems based on the platform Menlow were presented at the trade fair Embedded Word in Nuremburg [MCT08]. Via wants to launch a new CPU (Isaiah) in spring 2008, which is two times more powerful than the C7 and has a typical power consumption of less than 5 watt [HEN08]. The Isaiah should also have at minimum the computing power of the system ATH64_1.8. On the bottom of the scale of computing power for IA32 embedded systems there are the VIA Eden ESP with 300 MHz up to 1000 MHz and the AMD Geode LX with 433 MHz up to 600 MHz. Systems with such a CPU are mostly also limited in the data transfer rate of the I/O-interfaces e. g. IDE, SATA and USB interfaces.

9.1 Traditional boot process

9.1.1 Configuration

For the first analysis of the embedded systems the system EPIA-ML (see appendix A.3) and GEODE-LX (see appendix A.6) are available. The system GEODE-LX could not be tested because the driver for the integrated graphical card has some severe bugs [BUGa] [BUGb] so that the distribution Ubuntu 7.10 could not be installed with a X-Server.
As mass storage the flash memory TRACF4GB and the hard disk SEA3.5SATA500GB are used. The Linux distribution Ubuntu 7.10 is installed with all updates, which are available on the date of testing. Typically such a system with low computing power has a light weight Linux installed with an light weight desktop environment. A comparison of several desktop environments is described in [LMA08]. The distribution Ubuntu 7.10 is used to get the possibility to compare the results with the others of this work.

9.1.2 Measurement methods

Fbtt is used as described in chapter 3.1.3 and in another run a boot chart is generated, confer with chapter 3.1.2.

9.1.3 Results

Three runs are done with Fbtt for each configuration.

	Run 1 [s]	Run 2 [s]	Run 3 [s]	Mean [s]
SEA3.5SATA500GB	68.08	67.75	68.09	67.97
TRACF4GB	63.45	62.85	63.39	63.23

Table 9.1: Boot times of Ubuntu 7.10 beta (EPIA-ML) with several mass storages

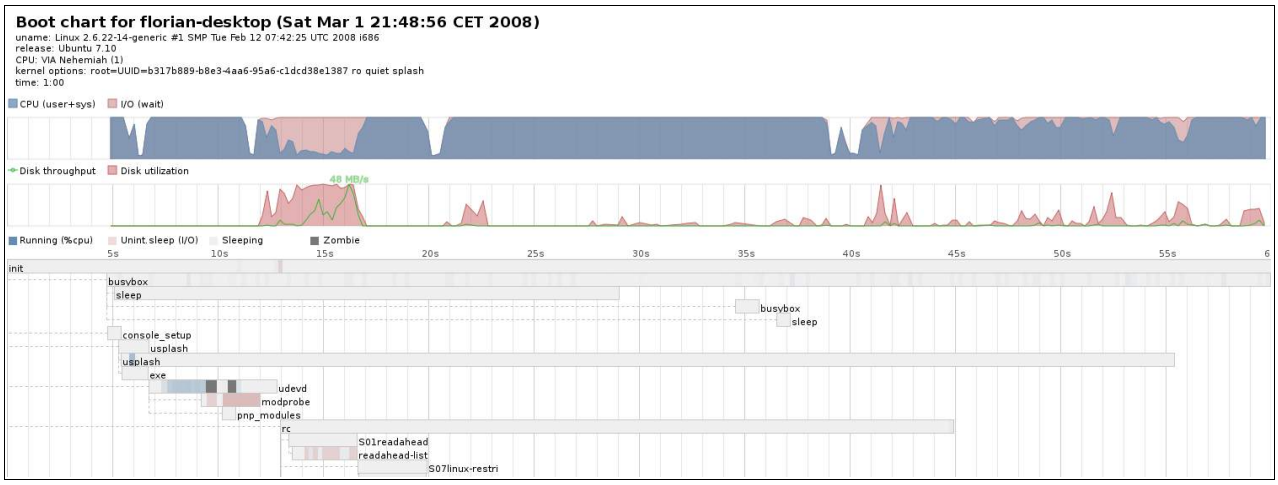


Figure 9.1: Abstract of the boot chart of Ubuntu 7.10 beta on System EPIA-ML with SEA3.5SATA500GB

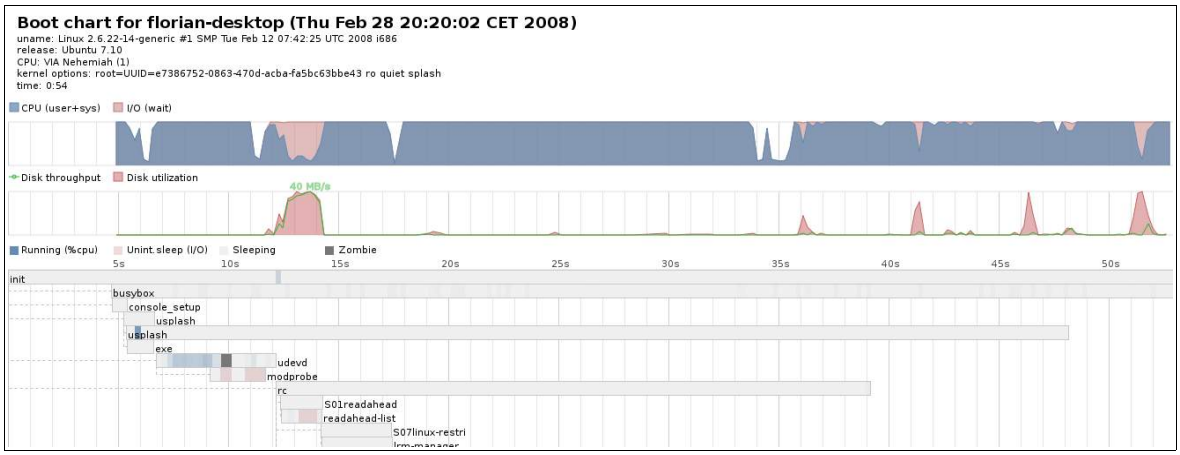


Figure 9.2: Abstract of the boot chart of Ubuntu 7.10 beta on System EPIA-ML with TRACF4GB

9.1.4 Discussion

The time for boot up Ubuntu 7.10 on an EPIAL-ML with the hard disk SEA3.5SATA500GB as mass storage needs with 68 seconds more than the twice the time of system ATH64X2_2.0 with SEA3.5SATA500GB, which needs only 28 seconds (confer chapter

6.1.3). The time gap between the usage of the compact flash memory TRACF4GB and the hard disk is on the EPIA-ML noticeable with four seconds.
The chart of Bootchart shows, that the CPU of the EPIA-ML is fully loaded the most time. The only way to reduce the boot time is to decrease the workload e. g. by deleting services. But more than the double of the time, which is gained on the system ATH64X2_2.0 (confer chapter 7.3) is not possible. A reduction of a few seconds by a boot time of more than 60 seconds is nowhere near enough. An alternative can be to use a special Distribution like Ubuntu mobile [UBM07], which is designed for systems with low computing power and few main memory. It was planed, that the first stable version should be available in October 2007, but the release date was shifted to the April 2008.

9.2 Boot up by loading image

9.2.1 Configuration

The system EPIA-ML and the hard disk SEA3.5SATA500GB is used as mass storage. A measurement series with the system GEODE-LX could not be done (confer chapter 9.1.1). The distribution Ubuntu 7.10 and the hibernate extension TuxOnIce are installed and used as described in chapter 8.3.1.1.

9.2.2 Measurement methods

The measurement is done with a stop watch and the analysis is done as described in chapter.

9.2.3 Results

	Desktop environment without dropped caches	Desktop environment with dropped caches	GDM without dropped caches	GDM with dropped caches
Used memory [MiB]	319.71	188.109	164.25	92.02
Free memory [MiB]	152.19	283.789	307.69	379.88
Buffers [MiB]	9.25	0.58	5.80	0.00
Cached [MiB]	165.08	41.54	81.59	16.53
Image size uncompressed [MiB]	266.64	140.74	117.16	49.59
Image size compressed [MiB]	142.98	69.64	64.03	22.35
Compression in percent	46	50	45	54
Write speed [MiB/s]	10	11	10	13
Read speed [MiB/s]	27	28	28	31
Time for read image [s]	9.88	5.03	4.18	1.60
Time until TuxOnIce starts [s]	13.21	13.21	13.21	13.21
Time [s]	34.70	29.35	28.03	23.99

Table 9.2: Values of the TuxOnIce boot on EPIA-ML

A description of the Table “TuxOnIce boot” is in the appendix chapter F.2.

9.2.4 Discussion

The loading of an images of a ready to use state of the display manager GDM without dropping the caches needs approximately 28 seconds. It is 40 seconds faster (approximately 60 percent) than the normal boot process which needs 68 seconds (see chapter 9.1.3). Therefore the advantage to load an image on a slower system is relatively larger compared to a system with more computing power like the ATH64X2_2.0. The ratios between the variants on EPIA-ML correspond to the ones of ATH64X2_2.0 (confer chapter 8.3.1.3). The conclusions of the chapter 8.3.1.4 can be adopted to systems with low computing power. But the result for the all in all times, by realizing the optimizations as described in chapter chapter 8.3.1.4, should be marginally longer.

10 Comparison between several firmwares

10.1 PC-BIOS

10.1.1 Configuration and measurement method

Two systems have to be evaluated. First the ATH64X2_2.0 and second the EPIA-ML.

The time, which is needed by the PC-BIOS depends on on the settings. Some settings do not really make a disadvantage in time, in contrast other need a lot of time. [PAR06c]
The largest consumers of time have been disabled when they are not needed (see table 10.1 and 10.2).

Feature	**Setting**
Cool & Quit	Enabled
Netboot	Disabled
RAID enabled	Disabled
Onboard Silicon SATA	Disabled
Onboard NV SATA	Enabled
IDE Channel 0	Disabled
IDE Channel 1	Disabled
SATA First Master	Auto
SATA Second Master	None
SATA Third Master	None
SATA Fourth	None
POST Check LAN Cable	Disabled
Speech IC Reporter	Disabled
Instant Music	Disabled
Onboard NV LAN	Enabled
Onboard Marvel LAN	Disabled
Quick Boot	Enabled
Boot Up Floppy Seek	Disabled
Full Screen Logo	Disabled

Table 10.1: PC-BIOS settings of ATH64X2_2.0

Feature	Setting
Quick Boot	Enabled
Boot Up Floppy Seek	Disabled
Full Screen Logo	Disabled
LAN Boot	Disabled

Table 10.2: PC-BIOS settings of EPIA-ML

The hard disk SEA3.5SATA500GB and the compact flash card TRACF4GB with SATA-to-CF-Adapter is used as mass storage. The EPIA-ML does not have any SATA interface, so that for the hard disk and the compact flash card an additional IDE-to-SATA-Adapter (see appendix C.2) is used.
The measurements are done with a stop watch. The power on of the system is the starting point and the prompt of the grub boot loader is the end point of the measurement. The measurement series with each system and mass storage is composed of a measurement with plugging the mass storage first to a local and second to a external power supply. This is done to find out what the influence of the spin up of the hard disk on the firmware start up is. The compact flash does not have any spin up time, because it has no moving parts inside. It is used as a reference and serves to exclude other influences. The external power supply is provided by a 5.25” external storage case.

10.1.2 Results

Three runs are done for each configuration.

ATH64X2_2.0

	Run 1 [s]	Run 2 [s]	Run 3 [s]	Mean [s]
SEA3.5SATA500GB	16.21	16.19	16.28	16.23
TRACF4GB	16.01	15.94	15.90	15.95

Table 10.3: Results with using internal power supply (PC-BIOS ATH64X2_2.0)

	Run 1 [s]	Run 2 [s]	Run 3 [s]	Mean [s]
SEA3.5SATA500GB	16.33	16.32	16.29	16.31
TRACF4GB	16.06	15.96	16.01	16.01

Table 10.4: Results with using external power supply (PC-BIOS ATH64X2_2.0)

EPIA-ML

	Run 1 [s]	Run 2 [s]	Run 3 [s]	Mean [s]
SEA3.5SATA500GB	11.21	11.12	11.28	11.20
TRACF4GB	11.30	11.10	11.18	11.19

Table 10.5: Results with using internal power supply (PC-BIOS EPIA-ML)

	Run 1 [s]	Run 2 [s]	Run 3 [s]	Mean [s]
SEA3.5SATA500GB	11.15	11.25	11.10	11.17
TRACF4GB	11.19	11.12	11.22	11.18

Table 10.6: Results with using external power supply (PC-BIOS EPIA-ML)

10.2 LinuxBios

10.2.1 Configuration and measurement method

The system EPIA-ML is used, because there was a ROM image available. An introduction to generate a LinuxBios image can be looked up in [PAR06].
The hard disk SEA3.5SATA500GB and the compact flash card TRACF4GB is used as mass storage. Furthermore the adapter are applied as described in chapter 10.1.1. The measuring is done manually with a stop watch. The power on of the system is the starting point and the error message “File not Found hda1:/vmlinuz-filo..” of the *filo* boot loader, after the identification of the mass storage by the boot loader, is the end point.
The output of the EPIA-ML system with the LinuxBios is done over the serial interface (COM) and is read out with the tool hyperterminal.
The measurement series is done with using a local and external power supply for the mass storages as described in chapter 10.1.1.

10.2.2 Results

Three runs are done for each configuration.

	Run 1 [s]	Run 2 [s]	Run 3 [s]	Mean [s]
SEA3.5SATA500GB	7.03	6.92	6.96	6.97
TRACF4GB	6.97	6.89	6.98	6.95

Table 10.7: Results with using internal power supply (LinuxBios EPIA-ML)

	Run 1 [s]	Run 2 [s]	Run 3 [s]	Mean [s]
SEA3.5SATA500GB	6.95	6.89	6.99	6.94
TRACF4GB	6.91	6.95	6.93	6.93

Table 10.8: Results with using external power supply (LinuxBios EPIA-ML)

10.3 EFI

10.3.1 Configuration and measurement method

As system with an EFI a Mac Mini is used. There is no tool available to change the configuration of EFI on the Mac Mini.
Nevertheless when the EFI module rEFIT 0.10. REFIT, a boot loader for the EFI firmware, has been added, the time between the boot up of the firmware and the operating system can be distinguished.
The the measurement starts with power on of the MAC-MINI and ends with the appearance of the logo of rEFIT. At that point rEFIT is not ready to use. The time is taken with a stop watch.
The MAC-MINI has been opened and modified for the measurements, therefore the following tests are possible to realize.
First, the internal DVD drive is cable-connected and a DVD or CD has not been inserted. Second, the internal DVD drive has been removed. Besides this, the measurements are done with and without an external power supply for the hard disk. Therefore the hard disk can be operated on the MAC-MINI with an external power supply (see appendix chapter D), a SATA-SATA-Adapter is needed (see appendix C.3), because the hard disk is connected to a SATA-Back-Plane by default and only a normal SATA data cable is not enough.

10.3.2 Results

Three runs are done for each configuration.

Configuration with	Run 1 [s]	Run 2 [s]	Run 3 [s]	Mean [s]
HDD Spin Up, DVD Drive	8.20	8.22	8.21	8.21
HDD Spin Up, without DVD Drive	7.61	7.64	7.64	7.63
HDD on external power supply, DVD Drive	6.08	6.10	6.03	6.07
HDD on external power supply, without DVD Drive	5.56	5.61	5.51	5.56

Table 10.9: Result of EFI evaluation

10.4 Discussion

In the case, that the evaluation of several firmwares is done on different systems, the result can not be compared directly. Currently this is not avoidable, because no system exists, which is supported by all the three firmwares. Although we can get some interesting results.

The results of the LinuxBios and PC-BIOS are rather the same compared to these of the student research project of Daniel Parthey [PAR06]. Also the boot time of the two tested PC-BIOS (ATH64X2_2.0 and EPIA-ML) differ with more than 30 percent (five seconds). The reason of this is, that the main board of the ATH64X2_2.0 has more components, which are initialized by the PC-BIOS. But there is also a difference between the results above and the ones of [PAR06]. In the student research project [PAR06] the LinuxBios was handicapped by the spin up of the hard disk with additional four seconds. This can not be confirmed with the evaluation above. Unfortunately the research of Mr. Parthey does not make any advice of the used model of the hard disk. A spin up time of a hard disk with much more than 6 seconds is unusual. Maybe the spindle motor of the hard disk used in the research [PAR06] has a stealthy fault, that results in longer spin up times.

The LinuxBios of the EPIA-ML saves four seconds (approximately 35 percent) compared to the PC-BIOS. The EFI of the MAC-MINI needs a similar amount of time like the EPIA-ML with LinuxBios, when the DVD drive of the MAC-MINI is disconnected and the internal power supply for the hard disk is used (the systems EPIA-ML and ATH64X2_2.0 are running always without a DVD drive). The EFI is handicapped by the spin up of the hard disk. It is approximately two seconds faster with using the external power supply and no waiting for spin up the hard disk, than with using the internal power supply and spinning up the hard disk by power on. Without the disadvantage of the spin up, the EFI of the MAC-MINI is a little bit faster (one and a half second, approximately 20 percent) than the

LinuxBios of the EPIA-ML. Whether the EFI or the LinuxBios is at last the faster one can not be answered, because the computing power of the both systems are very different.
With a time of seven seconds and less for executing the firmware (EFI and LinuxBios), the ambition of getting a complex system (with a full-grown Linux operating system included) in a ready to use state within 15 seconds is close to be realized.

11 Conclusion

In the thesis the boot up time of the Linux operation system is analyzed under different booting conditions. Along this it is examined which time is spent for each step of the boot process. Beyond this, the question is evaluated whether the booting times of new Linux distributions have increased despite new computer systems with more powerful processors are used.

To compare different boot up times several methods of measuring the boot up time of computer systems are presented and the accuracy and reliability of the results are shown. The tool Bootchart is helpful to analyze the boot process of Linux, but it is not qualified to get out the real boot up time. For this the tool Fbtt as an automatically measurement method and a stop watch as a manually method can be used. The method with a stop watch is absolutely without of any influences to the examined system and Fbtt impairs 0.04 seconds at the maximum, which is only 0.2 percent of the total boot time of Linux.

In the next chapter the myth of continuously rising boot up times is busted. Hence the most popular Linux distributions of the last two years are examined. The latest versions need the least time for the booting process than all other previous versions of each distribution in the last two years.

Chapter 5 goes more into the details of the latest distributions and their behavior by using several hard disks and different number of cores of the CPU. It is shown, that Debian 4 has a good time by using a scrawny System-V-Init process. The distribution of Ubuntu 7.10, which uses Upstart as init system, is fast in booting up, even thought it starts a lot of processes at boot time. The result of the single and dual core system differs only marginally. Therefore it can not use the computing power of the second core. In contrast OpenSUSE 10.3 uses often the second core, but reaches not the time of the test configuration of an single core system with using Ubuntu 7.10. Additionally openSUSE 10.3 is influenced more by a slower CPU (ATH64_1.8 instead ATH64X2_2.0) than a slower hard disk (notebook SAM2.5SATA100GB instead of a desktop SEA3.5SATA500GB).

The next research shows the results by using several local mass storages and a network storage. Expectedly the desktop hard disk SEA3.5SATA500GB is in the mean with 28 seconds faster than the notebook hard disk SAM2.5SATA100GB, which need 34 seconds at the boot process. A huge difference exists between the professional and low cost compact flash memory. The TRACF4GB (professional) reaches the best time in booting up with 21 seconds in the mean. In contrast, the TAKCF4GB (low cost) is significantly the slowest medium with 61 seconds in the mean. The boot process over a Gigabit Ethernet network does not result in an advantage in time. The client reaches the time of a system with SEA3.5SATA500GB as local mass storage only by reading out the data from the system buffer. The overhead of NFS and the latency of the Ethernet is to high. During the experiments the sneaking suspicion arises, that the spreading of the measurement values are larger with the usage of flash memory than with the desktop hard disk

SEA3.5SATA500GB, so that this phenomenon is examined too. The research results in, that the spreading of the TRACF4GB is really larger as the hard disk SEA3.5SATA500GB.
In chapter 7 Ubuntu 7.10 is tweaked to reduce the boot time. In the publication [THE07] the boot time of Ubuntu 7.04 Ultimate is reduced from 35 to 15 seconds (measurement method was Bootchart). The research in this work shows, that the well matched distribution Ubuntu 7.10 has only a marginally advantage of three seconds.
In the next chapter, a solution is examined, which does not follow the standard boot process scheme. Instead of rerunning the boot up process again and again, an image of an ready to use state is loaded. For the research is TuxOnIce used. The loading of an images of a ready to use state of the display manager GDM without dropping the caches is 12 seconds faster (more than 40 percent) than the normal boot process which needed 27 seconds. With the suggested optimizations the result can be much better.
In the chapter 9 on the embedded system EPIA-ML with IA32 architecture the traditional boot process and the boot up with loading an image of a ready to use state is tested. The computing power of this system is on the bottom of the scale of computing power for IA32 embedded systems. The time for boot up Ubuntu 7.10 on an EPIAL-ML with the hard disk SEA3.5SATA500GB as mass storage needs with 68 seconds more than twice of the time of the system ATH64X2_2.0 with SEA3.5SATA500GB, which needs only 28 seconds. To boot up with an image, the EPIA-ML is not as fast as the ATH64X2_2.0, but the EPIA-ML saves compared to the normal boot up approximately 60 percent (the ATH64X2_2.0 saves only 40 percent) of the time.
At last several firmwares are compared. The LinuxBios of the EPIA-ML saves four seconds (approximately 35 percent) compared to the PC-BIOS. The EFI of the MAC-MINI is a little bit faster (one and a half second, approximately 20 percent) than the LinuxBios of the EPIA-ML. Whether the EFI or the LinuxBios is at last the faster one can not be answered, because the computing power of the both systems are very different.

The result of this thesis is, that in the last two years it was done a lot to accelerate the normal boot up process of Linux and that by now only marginally optimization potential is left. Therefore it is very difficult, possibly even impossible, to realize a boot up time of 15 seconds included the time for executing the firmware. To load an image of a ready to use state from a non-volatile storage into the main memory and activate it, a complex system based on a full-grown Linux operating system, which was powered off before, could be ready to use in 15 seconds (a (U)EFI or a LinuxBios as firmware is adopted). Such a system can be realized in the near future.

12 Outlook

The matter of the firmware has to be evaluated again, when more systems as the x86-Mac computer are on the market, which have an (U)EFI as firmware. In all probability the times for booting up the (U)EFI will differ between the manufacturers like the results of measurement of the PC-BIOS (confer chapter 10.1.2). The deployment of the LinuxBios goes continuously forward, so that it should be evaluated too. Especially desirable is a comparison of an (U)EFI and LinuxBios on the same system. Today the LinuxBios starts up in a few seconds, but more systems has to be supported by the LinuxBios and the process of generating a LinuxBios for a system has to get more simple.

The results of getting a system in a ready to use state by the usage of TuxOnIce looks promising and a lot of potential of optimizations exist (confer chapter 8.3.1.4). The variant to load an image of a ready to use state with the aid of a tiny kernel is the one with the most potential. Besides, new measurement and analysis methods has to be find out, because the current ones do not work when the system is started with loading an image.

The usage of an LinuxBios or an (U)EFI as firmware in combination with loading an image and realizing some of the possible optimizations described in chapter 8.3.1.4, should let the all in all boot up time of an complex system (included a full-grown Linux distribution like Ubuntu) diminish to less than 15 seconds in the near future.

Although the loading of an image is significantly better than the normal boot up process, the normal boot up process may not be go disregarded, because a normal boot up can be avoided not in all cases, e. g. kernel update or after changing a not hot-plug-able device.

In the further future completely other possibilities will exist when non-volatile main memory is used (confer chapter 8.1). Then a system can be realized, which is ready to use in one second, although the system was disconnected from the power supply before. In the next time no such a system will come on the market, because up to now no manufacturer has announced such a product for the mainstream sector.

A Test systems

The data of “Processor”, “Chipset & Memory”, “System and Memory SPD” is read out with the software CPU-Z version 1.44 [CPUZ]. The information about the graphic interface and additional components are added manually.

The systems does not have any DVD Drive, when not otherwise described.

A.1 System ATH64_1.8

Processor(s)	
Number of processors	1
Number of cores	1 per processor
Number of threads	1 per processor
Name	AMD Athlon 64 3000+
Code Name	Winchester
Specification	AMD Athlon(tm) 64 Processor 3000+
Package	Socket 939
Family/Model/Stepping	F.F.0
Extended Family/Model	F.1F
Brand ID	4
Core Stepping	DH8-D0
Technology	90 nm
Core Speed	1809.4 MHz
Multiplier x Bus speed	9.0 x 201.0 MHz
HT Link speed	1005.2 MHz
Stock frequency	1800 MHz
Instruction sets	MMX (+), 3DNow! (+), SSE, SSE2, x86-64
L1 Data cache (per processor)	64 KBytes, 2-way set associative, 64-byte line size
L1 Instruction cache (per processor)	64 KBytes, 2-way set associative, 64-byte line size
L2 cache (per processor)	512 KBytes, 16-way set associative, 64-byte line size
Chipset & Memory	
Northbridge	NVIDIA nForce4 rev. A3
Southbridge	NVIDIA nForce4 MCP rev. A3
Graphic Interface	PCI-Express
PCI-E Link Width	x16
PCI-E Max Link Width	x16
Memory Type	DDR
Memory Size	512 MBytes
Memory Frequency	201.0 MHz (CPU/9)
CAS# Latency (tCL)	2.5 clocks
RAS# to CAS# (tRCD)	3 clocks
RAS# Precharge (tRP)	3 clocks
Cycle Time (tRAS)	8 clocks
Bank Cycle Time (tRC)	11 clocks
DRAM Idle Timer	16 clocks
Command Rate (CR)	1T

System	
System Manufacturer	System manufacturer
System Name	System name
System S/N	123456789000
Mainboard Vendor	ASUSTeK Computer INC.
Mainboard Model	A8N-SLI DELUXE
BIOS Vendor	Phoenix Technologies, LTD
BIOS Version	ASUS A8N-SLI DELUXE ACPI BIOS Revision 1016
BIOS Date	12/01/2005
Memory SPD	
Module 1	DDR, PC3200 (200 MHz), 512 MBytes, Corsair
Graphics Card	
Model	Sapphire ATI Radeon X600 pro 128 MB

Table A.1: Technical data of the system ATH64_1.8

A.2 System ATH64X2_2.0

Processor(s)	
Number of processors	1
Number of cores	2 per processor
Number of threads	2 per processor
Name	AMD Athlon 64 X2 3800+
Code Name	Manchester
Specification	AMD Athlon(tm) 64 X2 Dual Core Processor 3800+
Package	Socket 939
Family/Model/Stepping	F.B.1
Extended Family/Model	F.2B
Brand ID	5
Core Stepping	BH-E4
Technology	90 nm
Core Speed	2010.5 MHz
Multiplier x Bus speed	10.0 x 201.0 MHz
HT Link speed	1005.2 MHz
Stock frequency	2000 MHz
Instruction sets	MMX (+), 3DNow! (+), SSE, SSE2, SSE3, x86-64
L1 Data cache (per processor)	2 x 64 KBytes, 2-way set associative, 64-byte line size
L1 Instruction cache (per processor)	2 x 64 KBytes, 2-way set associative, 64-byte line size
L2 cache (per processor)	2 x 512 KBytes, 16-way set associative, 64-byte line size
Chipset & Memory	
Northbridge	NVIDIA nForce4 rev. A3
Southbridge	NVIDIA nForce4 MCP rev. A3
Graphic Interface	PCI-Express
PCI-E Link Width	x16
PCI-E Max Link Width	x16
Memory Type	DDR
Memory Size	1024 MBytes
Memory Frequency	201.0 MHz (CPU/10)

CAS# Latency (tCL)	2.5 clocks
RAS# to CAS# (tRCD)	3 clocks
RAS# Precharge (tRP)	3 clocks
Cycle Time (tRAS)	8 clocks
Bank Cycle Time (tRC)	11 clocks
DRAM Idle Timer	16 clocks
Command Rate (CR)	2T
System	
System Manufacturer	System manufacturer
System Name	System name
System S/N	123456789000
Mainboard Vendor	ASUSTeK Computer INC.
Mainboard Model	A8N-SLI DELUXE
BIOS Vendor	Phoenix Technologies, LTD
BIOS Version	ASUS A8N-SLI DELUXE ACPI BIOS Revision 1016
BIOS Date	12/01/2005
Memory SPD	
Module 1	DDR, PC3200 (200 MHz), 512 MBytes, Corsair
Module 2	DDR, PC3200 (200 MHz), 512 MBytes, Corsairhttp://www.opensuse.org/
Graphics Card	
Model	MSI nVidia Geforce 7600 GT

Table A.2: Technical data of the system ATH64X2_2.0

A.3 System EPIA-ML

Processor(s)	
Number of processors	1
Number of cores	1 per processor
Number of threads	1 per processor
Name	VIA C3
Code Name	Nehemiah
Specification	VIA Nehemiah
Package	Socket 370 CPGA
Family/Model/Stepping	6.9.8
Extended Family/Model	0.0
Core Stepping	C5P
Technology	0.13 um
Core Speed	666.6 MHz
Multiplier x Bus speed	5.0 x 133.3 MHz
Stock frequency	666 MHz
Instruction sets	MMX, SSE
L1 Data cache (per processor)	64 KBytes, 4-way set associative, 32-byte line size
L1 Instruction cache (per processor)	64 KBytes, 4-way set associative, 32-byte line size
L2 cache (per processor)	64 KBytes, 16-way set associative, 32-byte line size
Number of processors	1
Number of cores	1 per processor
Chipset & Memory	

Northbridge	VIA CLE266 (VT8623) rev. 00
Southbridge	VIA VT8235 rev. 00
Memory Type	
Memory Size	512 MBytes
Northbridge	VIA CLE266 (VT8623) rev. 00
Southbridge	VIA VT8235 rev. 00
System	
System Manufacturer	VIA Technologies, Inc.
System Name	VT8623-8235
System S/N	
Mainboard Vendor	
Mainboard Model	EPIA-ML
BIOS Vendor	Award Software International, Inc.
BIOS Version	6.00 PG
BIOS Date	07/12/2004
Memory SPD	
Module 1	DDR, PC3200 (200 MHz), 512 MBytes, unknown brand
Graphics Card	
Model	onboard

Table A.3: Technical data of the system EPIA-ML

A.4 System CORE2DUO_2.16

Processor(s)	
Number of processors	1
Number of cores	2 per processor
Number of threads	2 per processor
Name	Intel Mobile Core 2 Duo T7400
Code Name	Merom
Specification	Intel(R) Core(TM)2 CPU T7400 @ 2.16GHz
Package	Socket 479 mPGA
Family/Model/Stepping	6.F.6
Extended Family/Model	6.F
Core Stepping	B2
Technology	65 nm
Core Speed	2161.4 MHz
Multiplier x Bus speed	13.0 x 166.3 MHz
Rated Bus speed	665.0 MHz
Stock frequency	2166 MHz
Instruction sets	MMX, SSE, SSE2, SSE3, SSSE3, EM64T
L1 Data cache (per processor)	2 x 32 KBytes, 8-way set associative, 64-byte line size
L1 Instruction cache (per processor)	2 x 32 KBytes, 8-way set associative, 64-byte line size
L2 cache (per processor)	4096 KBytes, 16-way set associative, 64-byte line size
Chipset & Memory	
Northbridge	Intel i945PM rev. 03
Southbridge	Intel 82801GHM (ICH7-M/U) rev. B0
Graphic Interface	PCI-Express

PCI-E Link Width	x16
PCI-E Max Link Width	x16
Memory Type	DDR2
Memory Size	2048 MBytes
Memory Frequency	332.5 MHz (1:2)
CAS# Latency (tCL)	4.0 clocks
RAS# to CAS# (tRCD)	4 clocks
RAS# Precharge (tRP)	4 clocks
Cycle Time (tRAS)	15 clocks
Bank Cycle Time (tRC)	20 clocks
Northbridge	Intel i945PM rev. 03
Southbridge	Intel 82801GHM (ICH7-M/U) rev. B0
System	
System Manufacturer	COMPAL
System Name	HEL80C
System S/N	2056773501499
Mainboard Vendor	COMPAL
Mainboard Model	HEL8X
BIOS Vendor	COMPAL
BIOS Version	122B
BIOS Date	07/09/2007
Memory SPD	
Module 1	DDR2, PC2-5300 (333 MHz), 1024 MBytes, MCI Computer
Module 2	DDR2, PC2-5300 (333 MHz), 1024 MBytes, MCI Computer
Graphics Card	
Model	nVidia GeForce Go 7600

Table A.4: Technical data of the system CORE2DUO_2.16

A.5 System P4_3.2

Processor(s)	
Number of processors	1
Number of cores	1 per processor
Number of threads	2 per processor
Name	Intel Pentium 4
Code Name	Prescott
Specification	Intel(R) Pentium(R) 4 CPU 3.20GHz
Package	
Family/Model/Stepping	F.4.1
Extended Family/Model	F.4
Core Stepping	E0
Technology	90 nm
Core Speed	3192.1 MHz
Instruction sets	MMX, SSE, SSE2, SSE3
L1 Data cache (per processor)	16 KBytes, 8-way set associative, 64-byte line size
Trace cache (per processor)	12 Kuops, 8-way set associative
L2 cache (per processor)	1024 KBytes, 8-way set associative, 64-byte line size

Chipset & Memory	
Memory Type	
Memory Size	2048 MBytes
System	
System Manufacturer	MEDIONPC
System Name	MS-7046
System S/N	
Mainboard Vendor	MICRO-STAR INTERNATIONAL CO., LTD
Mainboard Model	MS-7046
BIOS Vendor	Phoenix Technologies, LTD
BIOS Version	6.00 PG
BIOS Date	01/10/2005
Memory SPD	
Graphics Card	
Model	ATI Radeon X600

Table A.5: Technical data of the system P4_3.2

A.6 System GEODE-LX

Processor(s)	
Number of processors	1
Number of cores	1 per processor
Number of threads	1 per processor
Name	AMD Geode LX
Code Name	Castle
Specification	Geode(TM) Integrated Processor by AMD PCS
Package	Socket 481 BGU
Family/Model/Stepping	5.A.2
Extended Family/Model	5.A
Core Stepping	
Technology	0.13 um
Core Speed	498.0 MHz
Multiplier x Bus speed	15.0 x 33.2 MHz
Rated Bus speed	66.4 MHz
Instruction sets	MMX (+), 3DNow! (+)
L1 Data cache (per processor)	64 KBytes, 16-way set associative, 32-byte line size
L1 Instruction cache (per processor)	64 KBytes, 16-way set associative, 32-byte line size
L2 cache (per processor)	128 KBytes, 4-way set associative, 32-byte line size
Number of processors	1
Chipset & Memory	
Southbridge	AMD ID2090 rev. 02
Memory Type	
Memory Size	480 MBytes
System	

System Manufacturer	
System Name	
System S/N	
Mainboard Vendor	
Mainboard Model	AMD-LX800
BIOS Vendor	Phoenix Technologies, LTD
BIOS Version	6.00 PG
BIOS Date	11/09/2006
Memory SPD	
Graphics Card	
Model	integrated

Table A.6: Technical data of the system GOEDE-LX

A.7 System MAC-MINI

The information of the MAC-MINI is cut out from the system information of the operating system OS-X.

Processor(s)	
Number of processors	1
Number of cores	2 per processor
Name	Intel Core Duo
Core Speed	1666 MHz
Chipset & Memory	
Northbridge	-
Southbridge	-
Memory Type	DDR2
Memory Size	512 MBytes
Memory Frequency	667.0 MHz
System	
System Manufacturer	Apple
System Name	Macmini1,1
System S/N	YM63735NW0A
Boot-ROM-Version	MM11.0055.B08
SMC-Version	1.3f4
Memory SPD	
Module 1	DDR2, (667 MHz), 256 MBytes
Module 2	DDR2, (667 MHz), 256 MBytes
Graphics Card	
Model	GMA950

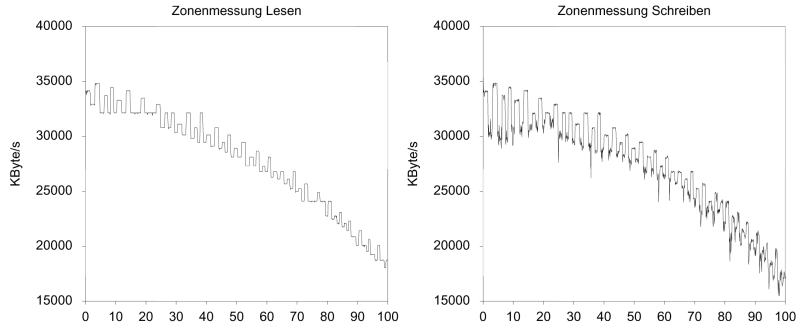
Table A.7: Technical data of the system MAC-MINI

B Technical data of mass storages

B.1 SAM2.5SATA100GB

SAMSUNG HM100JI

Mapping/Kapazität:	CHS=(12161/255/63), insgesamt 195366465 Sektoren = 95394 MByte
Blockgrößen:	physikalisch: 1*512 Byte, Dauertransferrate: 128 Sektoren (64.0 KByte)
Interface-Transferrate (bei 0.0%):	Ungebremst: 33.4 MByte/s, Verzögert (2.06 ms): 75.0 MByte/s, "Coretest": 91.7 MByte/s
Dauertransferrate Lesen:	Mittel 27038.4, Min 18044.1, Max 34827.1 [KByte/s]
Dauertransferrate Schreiben:	Mittel 26358.9, Min 15470.3, Max 35615.8 [KByte/s]
Zugriffszeit ganze Platte:	Mittel 13.80 (18.43r/9.17w, Min 4.81r/3.43w, Max 43.40r/22.44w) [ms]
Zugriffszeit erste 504 MByte:	Mittel 7.49 (9.93r/5.05w, Min 0.22r/0.29w, Max 33.89r/22.44w) [ms]



Anwendungsprofile [KByte/s]

Swappen: 8502.0
Installieren: 12836.8
Word: 17516.6
Photoshop: 13156.0
Kopieren: 55092.4
F-Prot: 7072.3
Anwendungsindex: 14628.5

ATA-Platte: SAMSUNG HM100JI
Seriennr.: S0EDJ10LA07999
Firmware: YH100-19
Standard-Version: ATA/ATAPI-7
Puffergröße: 8192 KByte
Unterstützte UDMA-Modi: 0 1 2 3 4 5 6 7
UDMA-Mode 7 aktiviert.
Kapazität (28-Bit-Adressierung): 195371568 Sektoren (95396.3 MByte)
Kapazität (48-Bit-Adressierung): 195371568 Sektoren (95396.3 MByte)
Akustik-Management vorhanden, aber abgeschaltet.
IDENTIFY DEVICE information:

```
  0 (0x00): 045a 3fff c837 0010 8856 022a 003f ffff 0000 0000
 10 (0x0a): 5330 4544 4a31 304c 4130 3739 3939 2020 2020 2020
 20 (0x14): 0003 4000 0004 5948 3130 302d 3139 5341 4d53 554e
 30 (0x1e): 4720 484d 3130 304a 4920 2020 2020 2020 2020 2020
 40 (0x28): 2020 2020 2020 2020 2020 2020 2020 8010 0000 2f00
 50 (0x32): 4000 0200 0200 0007 3fff 0010 003f fc10 00fb 0101
 60 (0x3c): 2230 0ba5 0000 0007 0003 0078 0078 0078 0078 0000
 70 (0x46): 0000 0000 0000 0000 0000 001f 0702 0000 004c 0040
 80 (0x50): 00fe 001e 746b 7f09 6023 3c69 3c01 6003 80ff 0023
 90 (0x5a): 0023 0080 fffe 0000 fe00 0000 0000 0000 0000 0000
100 (0x64): 2230 0ba5 0000 0000 0000 0000 0000 0000 50f0 0000
110 (0x6e): 0c10 7999 0000 0000 0000 0000 0000 0000 0000 4012
120 (0x78): 4010 0000 0000 0000 0000 0000 0000 0000 0021 0000
130 (0x82): 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
140 (0x8c): ffff 0400 1700 0000 0000 9a00 0300 2400 0000 0000
150 (0x96): 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
160 (0xa0): 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
170 (0xaa): 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
180 (0xb4): 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
190 (0xbe): 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
200 (0xc8): 0000 0000 0000 0000 0000 0000 003f 0000 0000 0000
210 (0xd2): 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
220 (0xdc): 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
230 (0xe6): 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
240 (0xf0): 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
250 (0xfa): 0000 0000 0000 0000 0000 55a5
```

Timerauflösung: 0.279 µs, 3.580 MHz, Timerstatistik: 131857594 Aufrufe, min 1.68 µs, mittel 7721.37 µs, max 6317145.06 µs
Kommandozeile: h2benchw.exe 1 -a -w SAMSUNGHM100JI -!
Testbeginn: 06.10.07 11:42:51, Testversion: $Id: h2bench.c,v 3.6 2002/10/31 14:06:35 bo Exp $/Win32, Dateiname 'SAMSUNGHM100JI'

Figure B.1: Output of h2benchw for SAM2.5SATA100GB

Manufacturer	Model Number	Interface	Form Factor	Capacity	CQ	RPM	Spin Up Time
Samsung	HM100JI	SATA-I	2.5”	100 GB	yes	5400	n. s.

Table B.1: Technical data of SAM2.5SATA100GB

B.2 SEA3.5SATA500GB

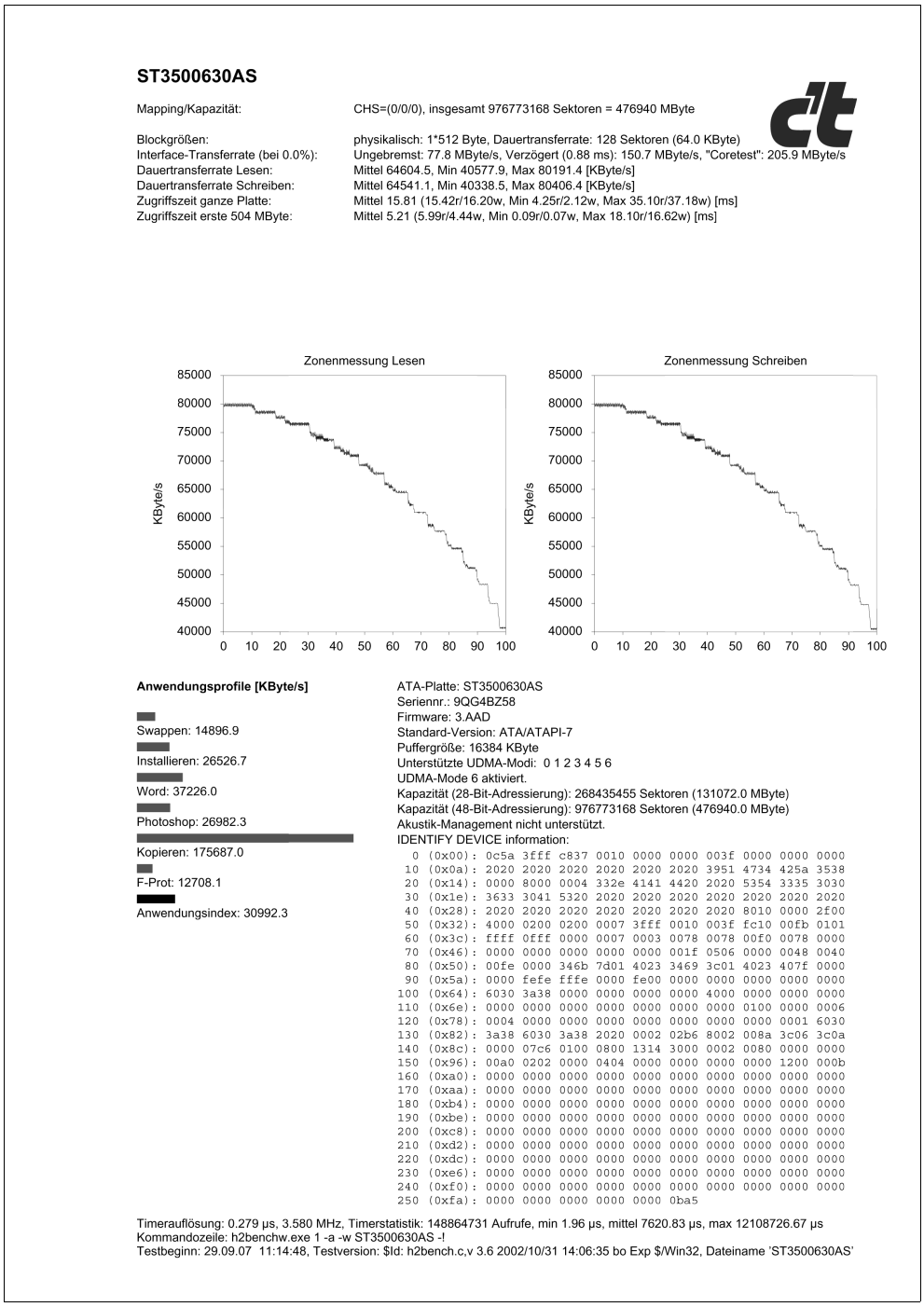


Figure B.2: Output of h2benchw for SEA3.5SATA500GB

Manufacturer	Model Number	Interface	Form Factor	Capacity	CQ	RPM	Spin Up Time
Seagate	ST3500630AS	SATA-II	3.5”	500 GB	yes	7200	n. s.

Table B.2: Technical data of SEA3.5SATA500GB

B.3 TRACF4GB

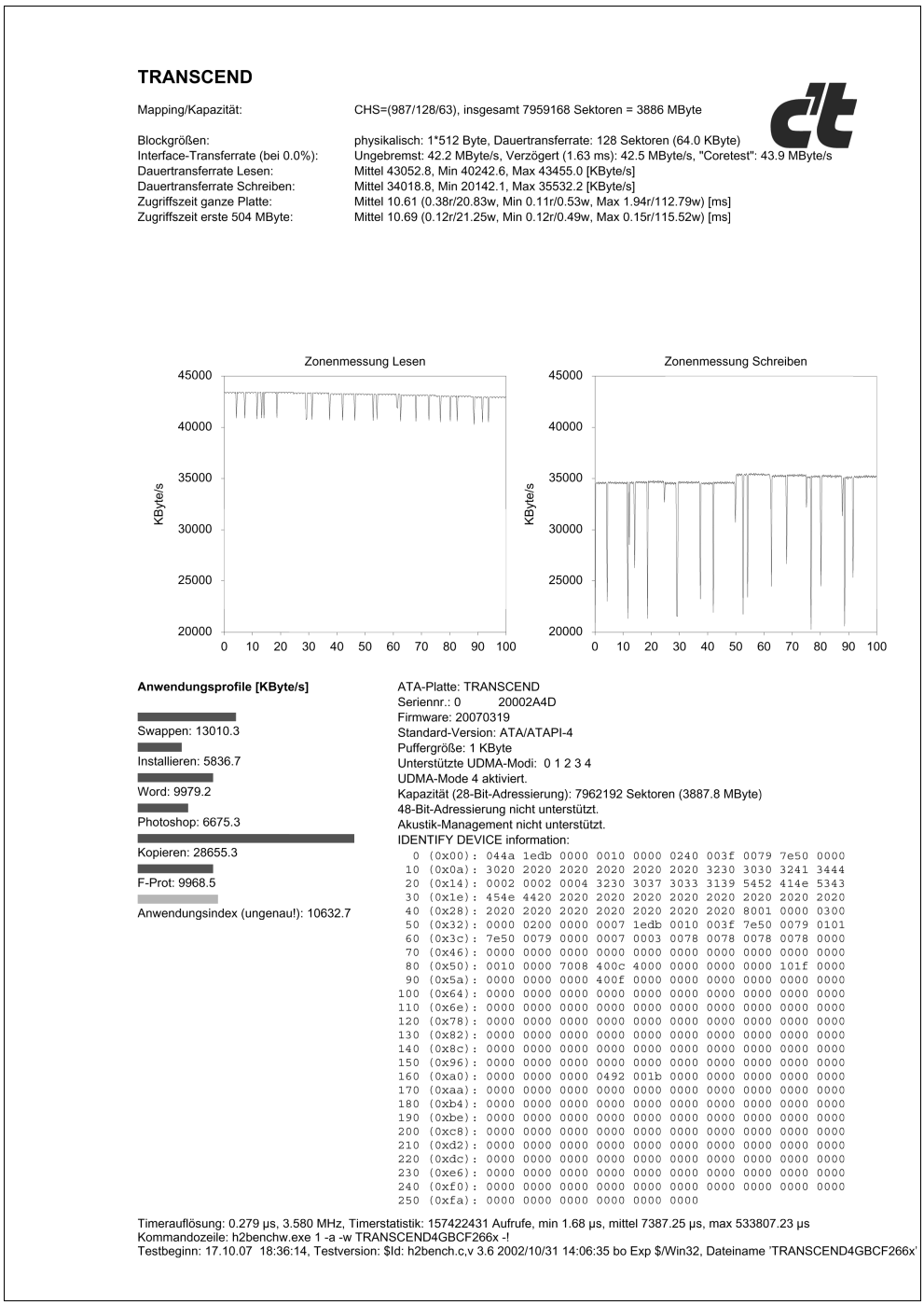


Figure B.3: Output of h2benchw for TRACF4GB

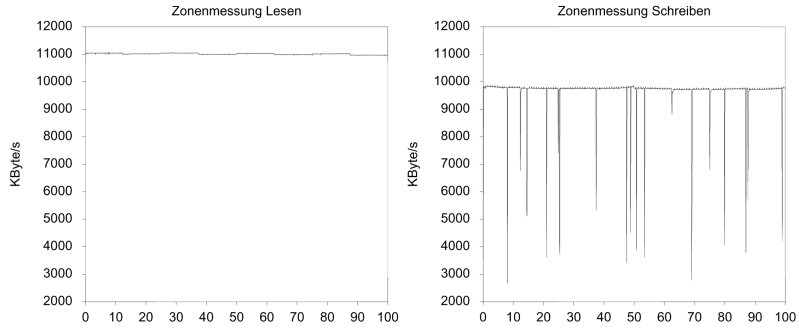
Manufacturer	Model Number	Interface	Form Factor	Capacity	RPM	Spin Up Time
Transcend	TS4GCF266	CF	CF type I	4 GB	-	-

Table B.3: Technical data of TRACF4GB

B.4 TAKCF4GB

SMI MODEL

Mapping/Kapazität:	CHS=(983/128/63), insgesamt 7926912 Sektoren = 3871 MByte
Blockgrößen:	physikalisch: 1*512 Byte, Dauertransferrate: 128 Sektoren (64.0 KByte)
Interface-Transferrate (bei 0.0%):	Ungebremst: 10.8 MByte/s, Verzögert (6.38 ms): 10.8 MByte/s, "Coretest": 11.2 MByte/s
Dauertransferrate Lesen:	Mittel 11014.1, Min 10953.3, Max 11076.8 [KByte/s]
Dauertransferrate Schreiben:	Mittel 9474.2, Min 2417.1, Max 9877.2 [KByte/s]
Zugriffszeit ganze Platte:	Mittel 182.33 (0.84r/363.82w, Min 0.17r/0.74w, Max 2.05r/726.42w) [ms]
Zugriffszeit erste 504 MByte:	Mittel 179.91 (0.20r/359.62w, Min 0.18r/0.70w, Max 0.23r/723.53w) [ms]



Anwendungsprofile [KByte/s]

Swappen: 2309.9
Installieren: 731.1
Word: 1250.2
Photoshop: 1064.7
Kopieren: 4702.1
F-Prot: 1380.8
Anwendungsindex (ungenau!): 1563.3

ATA-Platte: SMI MODEL
Seriennr.: SMI 0000083F
Firmware: 20070312
Puffergröße: 1 KByte
Gerät unterstützt kein Ultra-DMA!
Kapazität (28-Bit-Adressierung): 7928928 Sektoren (3871.5 MByte)
WARNUNG: Informationen in Word 82 bis 84 ungueltig
IDENTIFY DEVICE information:

```
  0 (0x00): 044a 1eba 0000 0010 0000 0240 003f 0078 fc60 0000
 10 (0x0a): 534d 4920 2020 2020 2020 2020 3030 3030 3038 3346
 20 (0x14): 0002 0002 0004 3230 3037 3033 3132 534d 4920 4d4f
 30 (0x1e): 4445 4c20 2020 2020 2020 2020 2020 2020 2020 2020
 40 (0x28): 2020 2020 2020 2020 2020 2020 2020 8001 0000 0300
 50 (0x32): 0000 0200 0000 0003 1eba 0010 003f fc60 0078 0101
 60 (0x3c): fc60 0078 0000 0407 0003 0078 0078 0078 0078 0000
 70 (0x46): 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
 80 (0x50): 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
 90 (0x5a): 0000 0000 0000 400f 0000 0000 0000 0000 0000 0000
100 (0x64): 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
110 (0x6e): 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
120 (0x78): 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
130 (0x82): 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
140 (0x8c): 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
150 (0x96): 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
160 (0xa0): 9096 0000 0000 0492 001b 0000 0000 0000 0000 0000
170 (0xaa): 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
180 (0xb4): 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
190 (0xbe): 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
200 (0xc8): 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
210 (0xd2): 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
220 (0xdc): 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
230 (0xe6): 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
240 (0xf0): 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
250 (0xfa): 0000 0000 0000 0000 0000 0000
```

Timerauflösung: 0.279 µs, 3.580 MHz, Timerstatistik: 107839970 Aufrufe, min 2.23 µs, mittel 8725.56 µs, max 1641655.85 µs
Kommandozeile: h2benchw.exe 1 -a -w takeMS4GBCF -!
Testbeginn: 19.12.07 11:27:30, Testversion: $Id: h2bench.c,v 3.6 2002/10/31 14:06:35 bo Exp $/Win32, Dateiname 'takeMS4GBCF'

Figure B.4: Output of h2benchw for TAKCF4GB

Manufacturer	Model Number	Interface	Form Factor	Capacity	RPM	Spin Up Time
TakeMS	MS4096CFLAH010	CF	CF type I	4 GB	-	-

Table B.4: Technical data of TAKCF4GB

B.5 MAC-MINI HDD

Manufacturer	Model Number	Interface	Form Factor	Capacity	RPM	Spin Up Time
Fujitsu	MHV2060BHPL	SATA-I	2.5”	60 GB	5400	4 sec.

Table B.5: Technical data of MAC-MINI hard disk

C Adapter (SATA / IDE / CF)

C.1 SATA-CF-Adapter

The SATA-CF-Adapter is used for connecting a compact flash card to a SATA-Interface.

Figure C.1: SATA-CF-Adapter (DeLOCK 91623)

C.1.1 Troubleshooting with the SATA-CF-Adapter

Only two compact flash cards (TRACF4GB see chapter B.3 and TAKCF4GB see chapter B.4) of a couple of cards do work with the SATA-CF-Adapter (see figure C.1) in combination with the nVidia nForce4-SLI chip set, which is integrated in the systems ATH64_1.8 and ATH64X2_2.0 (see chapter A.1 and A.2).
The following impacts can occur when the compact flash card does not work stable with the chip set together:
The compact flash card is not detected, slows down the system or produces a system crash with a blue screen by using the operating system Microsoft Windows.

The difference between the working and non-working cards is, that the working ones are marked as fixed disk and the UDMA mode is supported.
With another system, based on a Pentium 4 and a chip set of Intel, all cards do work without any negative influence.
A request to the distributor DeLOCK regarding the SATA-CF-Adapter results in the answer, that there are no problems known. In the data sheet [CON06] of a SATA-CF-Adapter, which is similar in design and is sold by Conrad Electronic SE, the hint is given, that the adapter only works with Intel and specific Silicon Image chip sets.

C.2 IDE-SATA-Adapter

The IDE-SATA-Adapter is used for connecting a SATA device to an IDE-Interface.

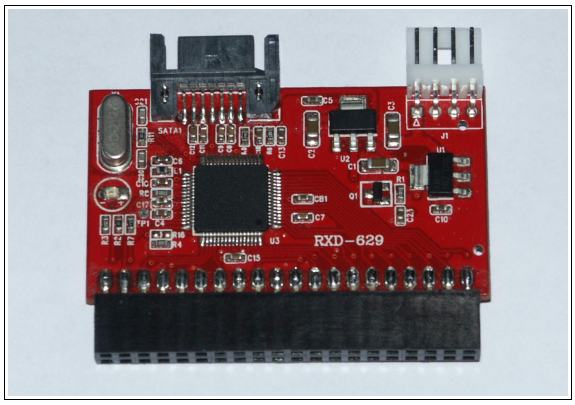


Figure C.2: IDE-SATA-Adapter (Fibrionic Model No: CK-0019C Ro)

C.3 SATA-SATA-Adapter

The IDE-SATA-Adapter is used for connecting a SATA device with a SATA data cable to a SATA-Backplane. Therefore the SATA-Device, which is connected to the backplane can be run on an external power supply. This adapter is created by cutting out from a SATA-to-SATA bracket, because such a adapter could not be purchased.

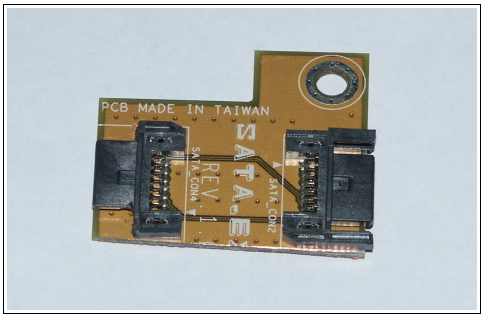


Figure C.3: SATA-SATA-Adapter

D Modification of the Mac Mini

To check if the spin up of the hard disk is slown down by the (U)EFI firmware of the Mac Mini, the case is opened as shown in the video [YOU07]. After this the hard disk is removed and reconnected with a SATA-SATA-Adapter to be able to run the device on an external power supply. The figure D.1 shows the configuration of the opened Mac Mini, using an external power supply for the hard disk and the removed DVD drive. On figure D. 2 the configuration can be seen, with the cable-connected DVD drive and the hard disk using an external power supply.

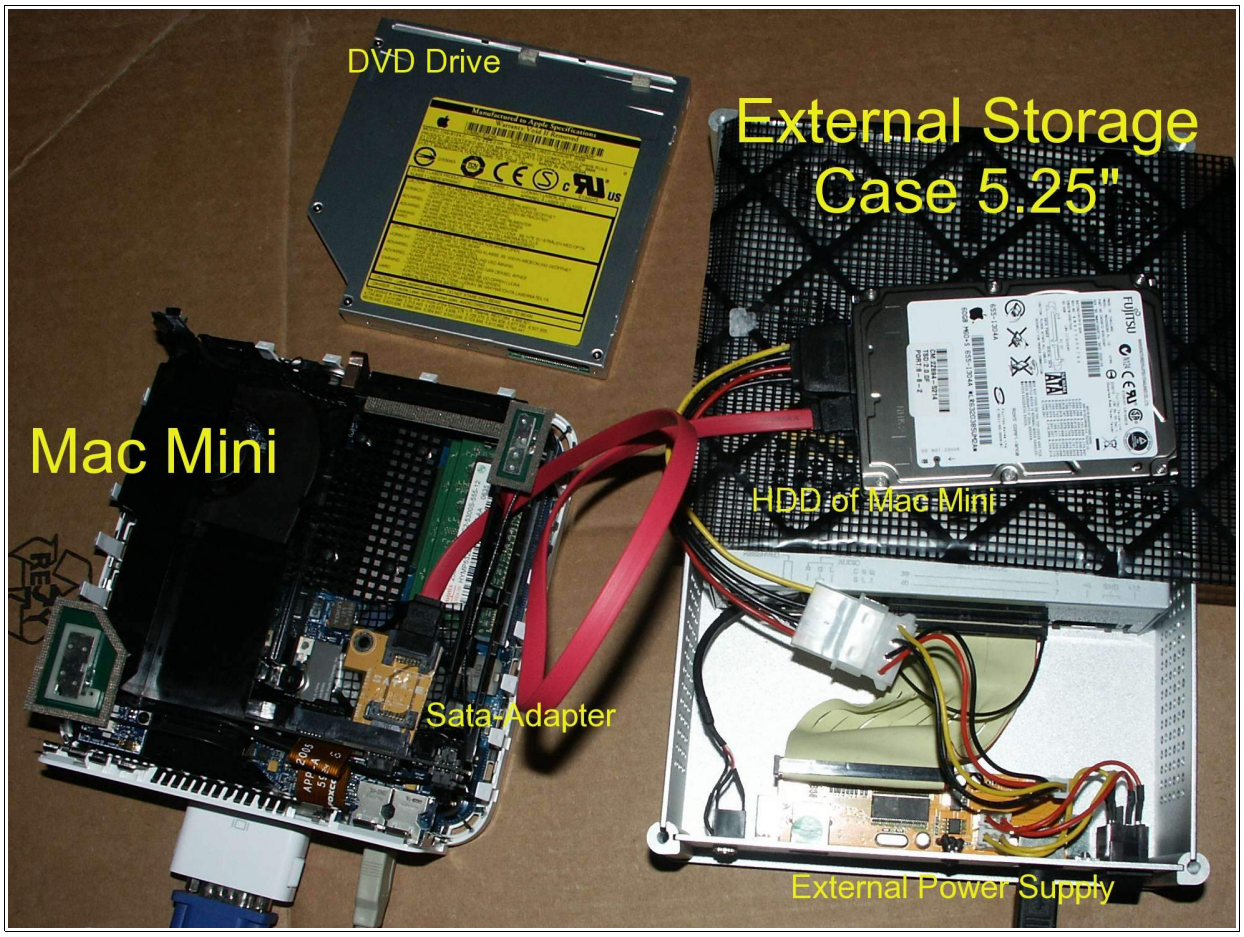


Figure D.1: Opened Mac Mini with the removed DVD drive and the external power supply for the hard disk

Figure D.2: Opened Mac Mini with the cable-connected DVD drive and the external power supply for the hard disk

E Charts of Bootchart

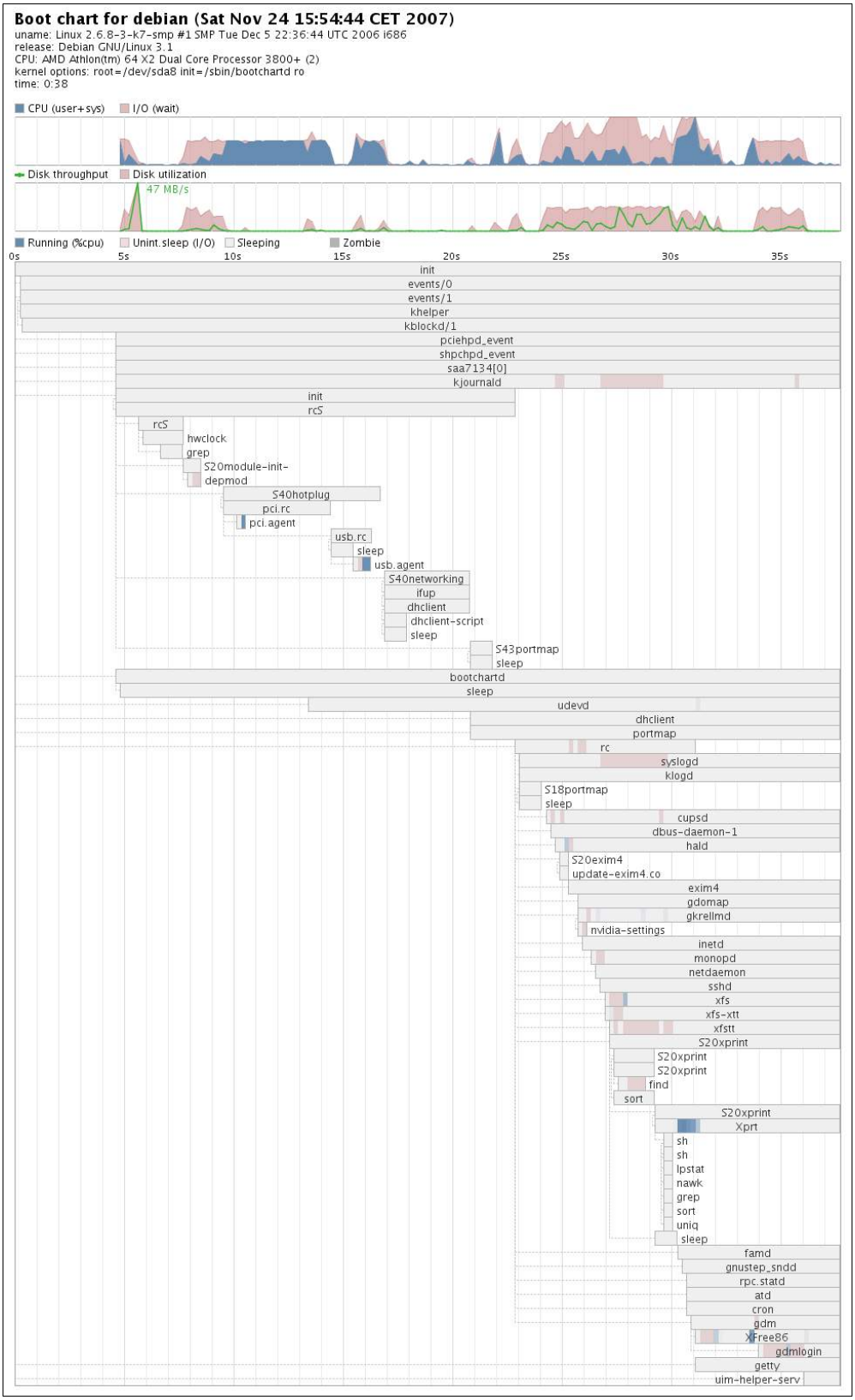


Figure E.1: Boot chart of boot process of Debian 3.1 on System ATH64X2_2.0 with SEA3.5SATA500GB

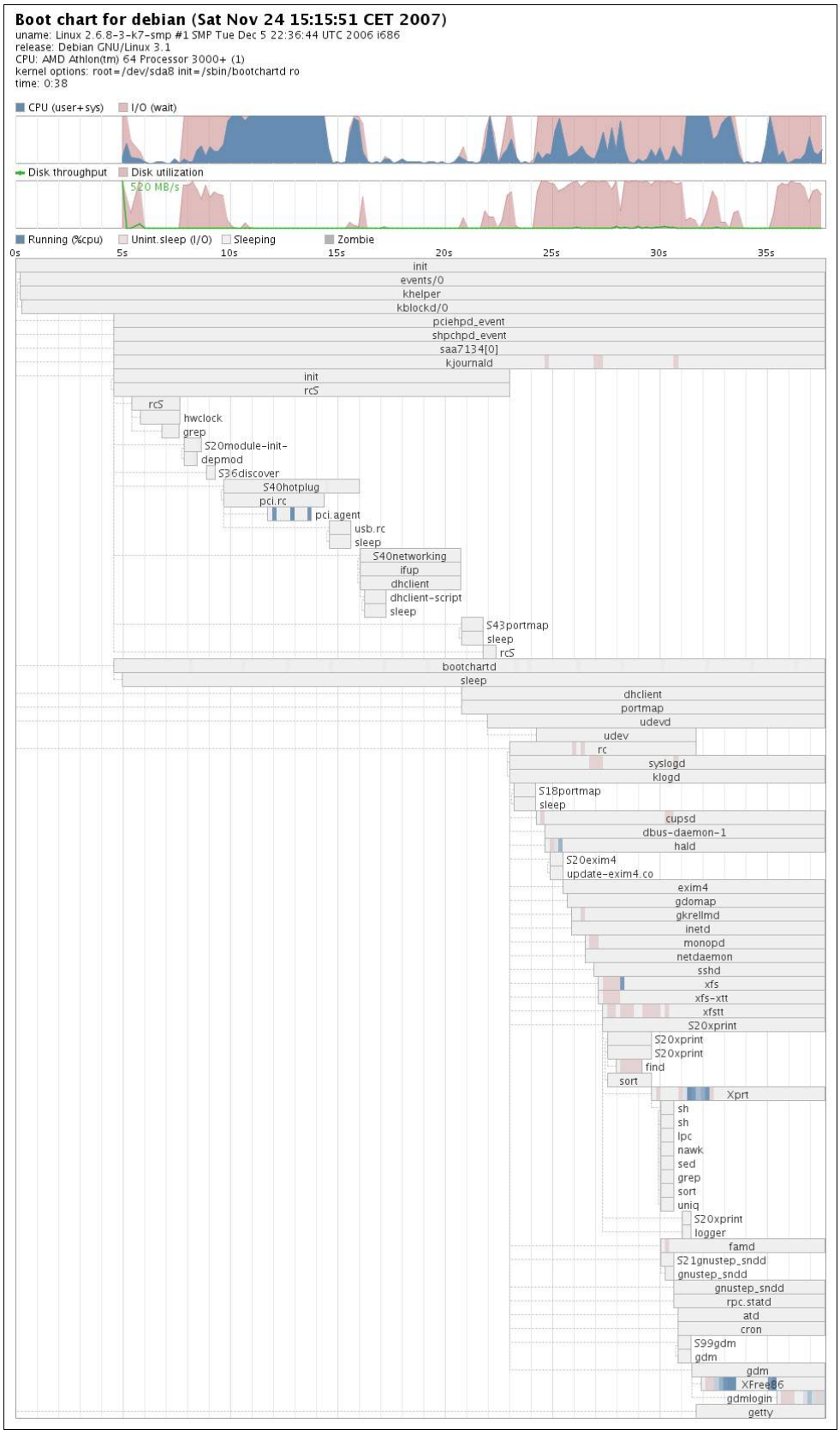


Figure E.2: Boot chart of boot process of Debian 3.1 on System ATH64_1.8 with SEA3.5SATA500GB

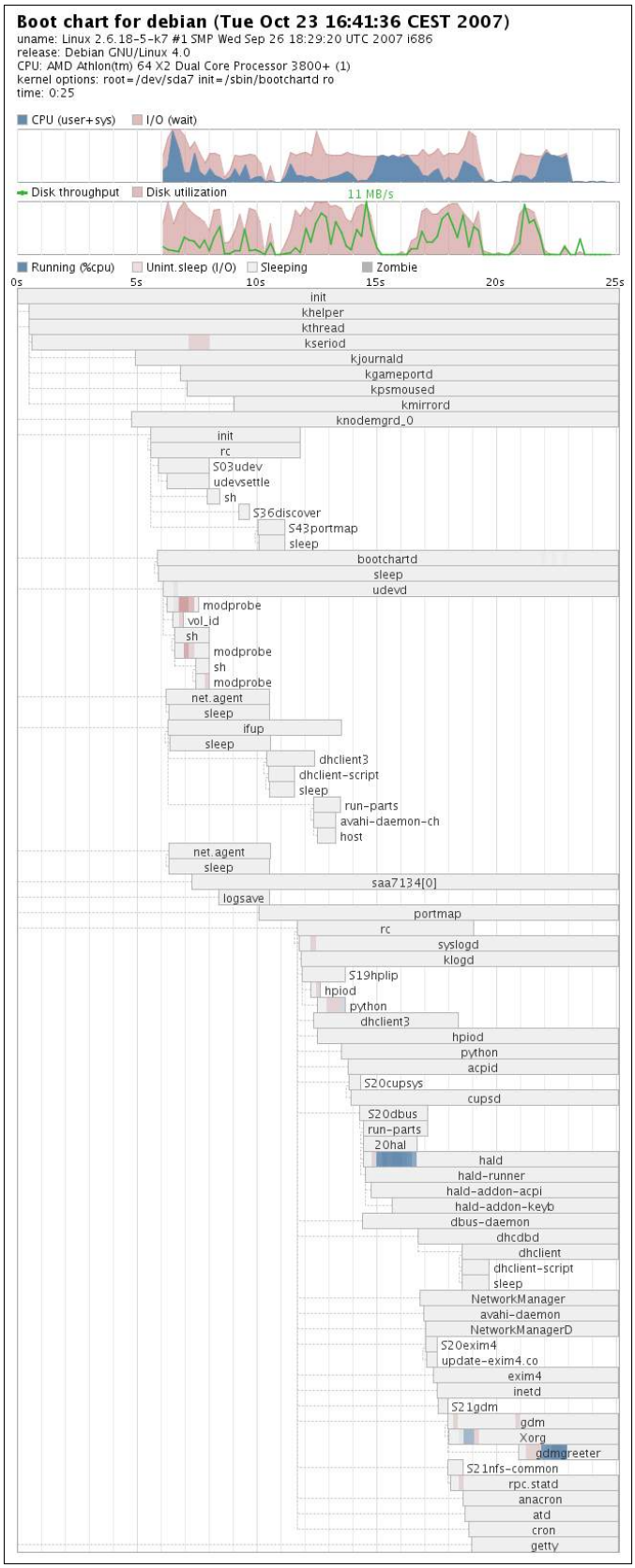


Figure E.3: Boot chart of boot process of Debian 4.0 on System ATH64X2_2.0 with SEA3.5SATA500GB

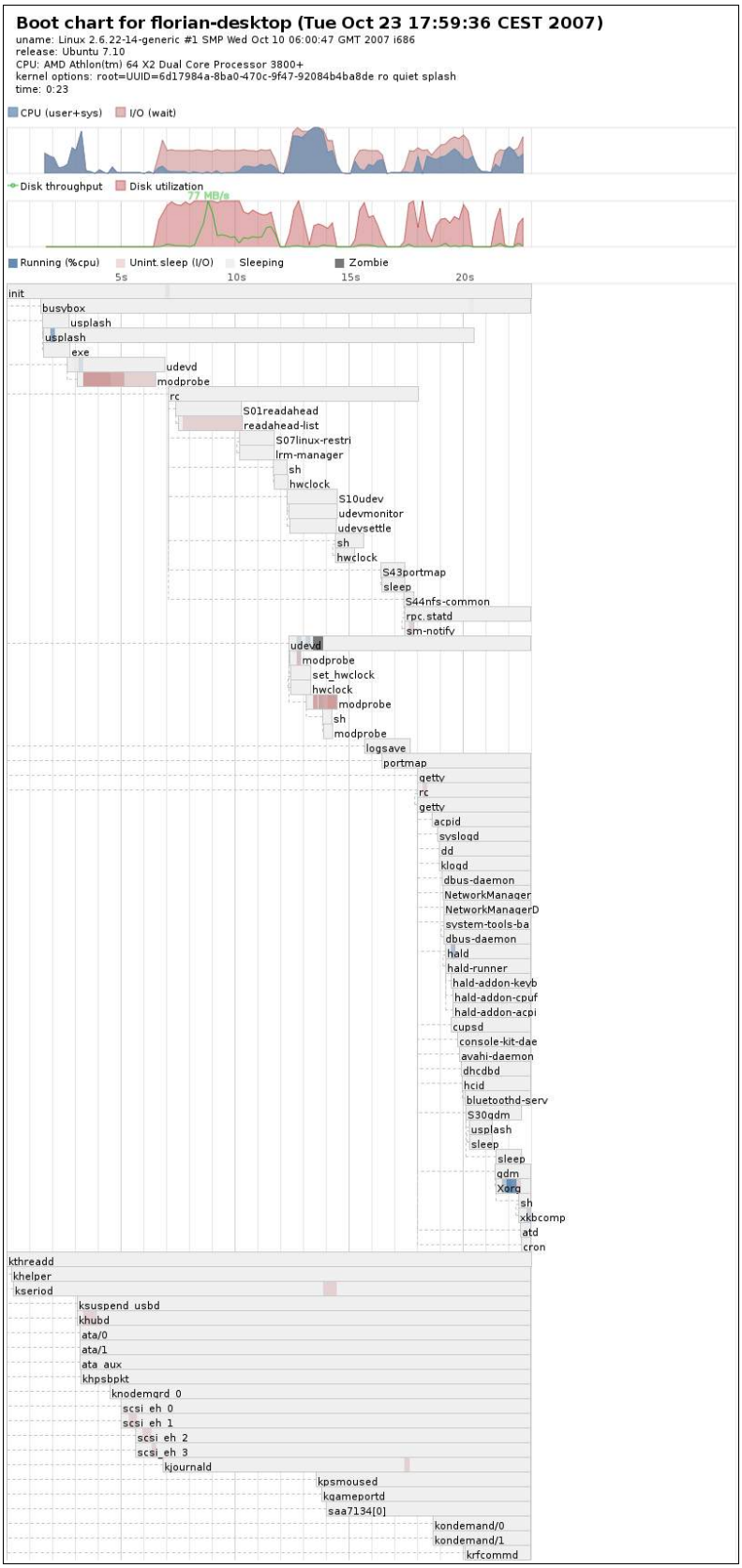


Figure E.4: Boot chart of boot process of Ubuntu 7.10 beta on System ATH64X2_2.0 with SEA3.5SATA500GB

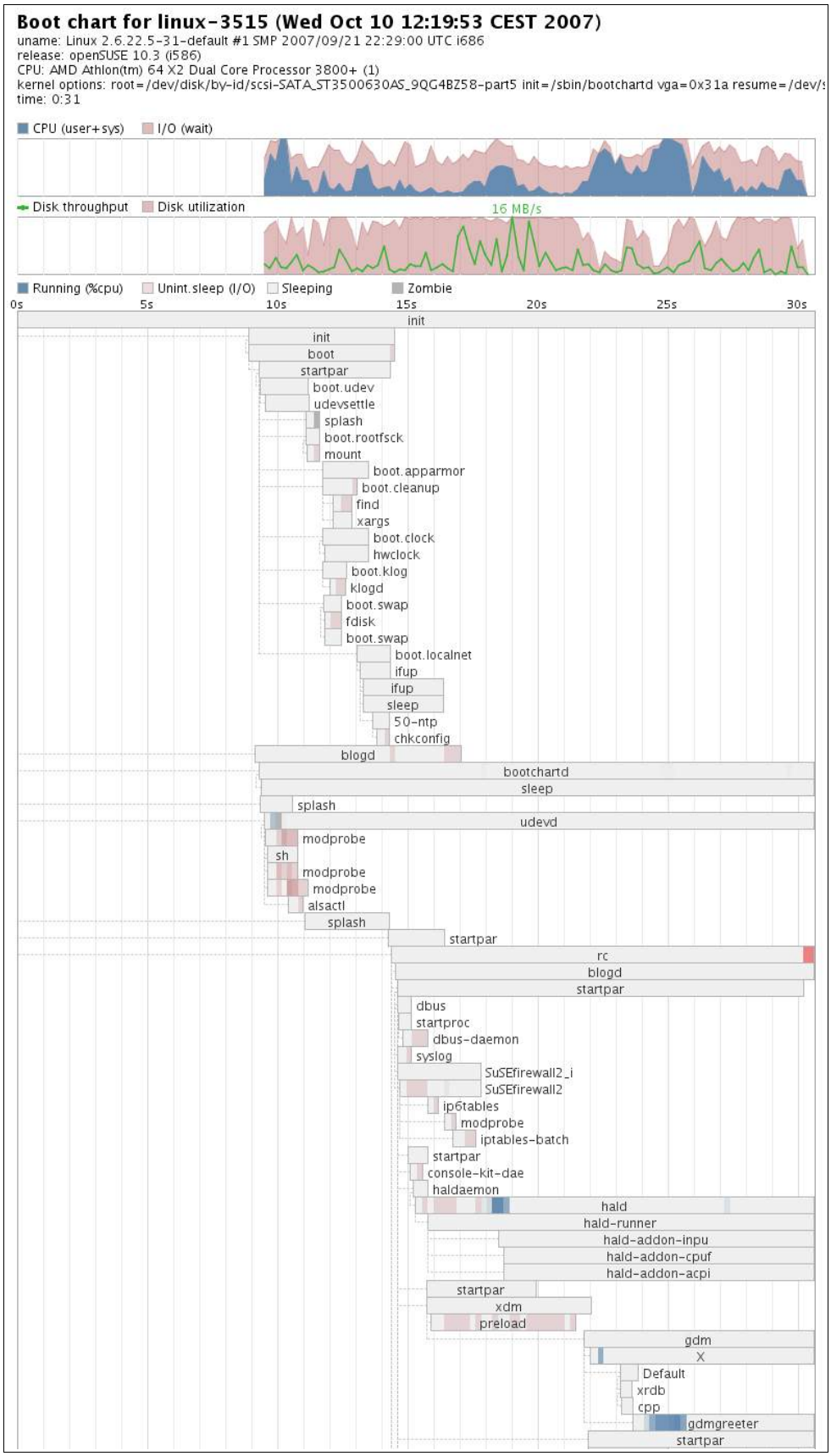


Figure E.5: Boot chart of boot process of openSUSE 10.3 RC1 on System ATH64X2_2.0 with SEA3.5SATA500GB (Part 1)

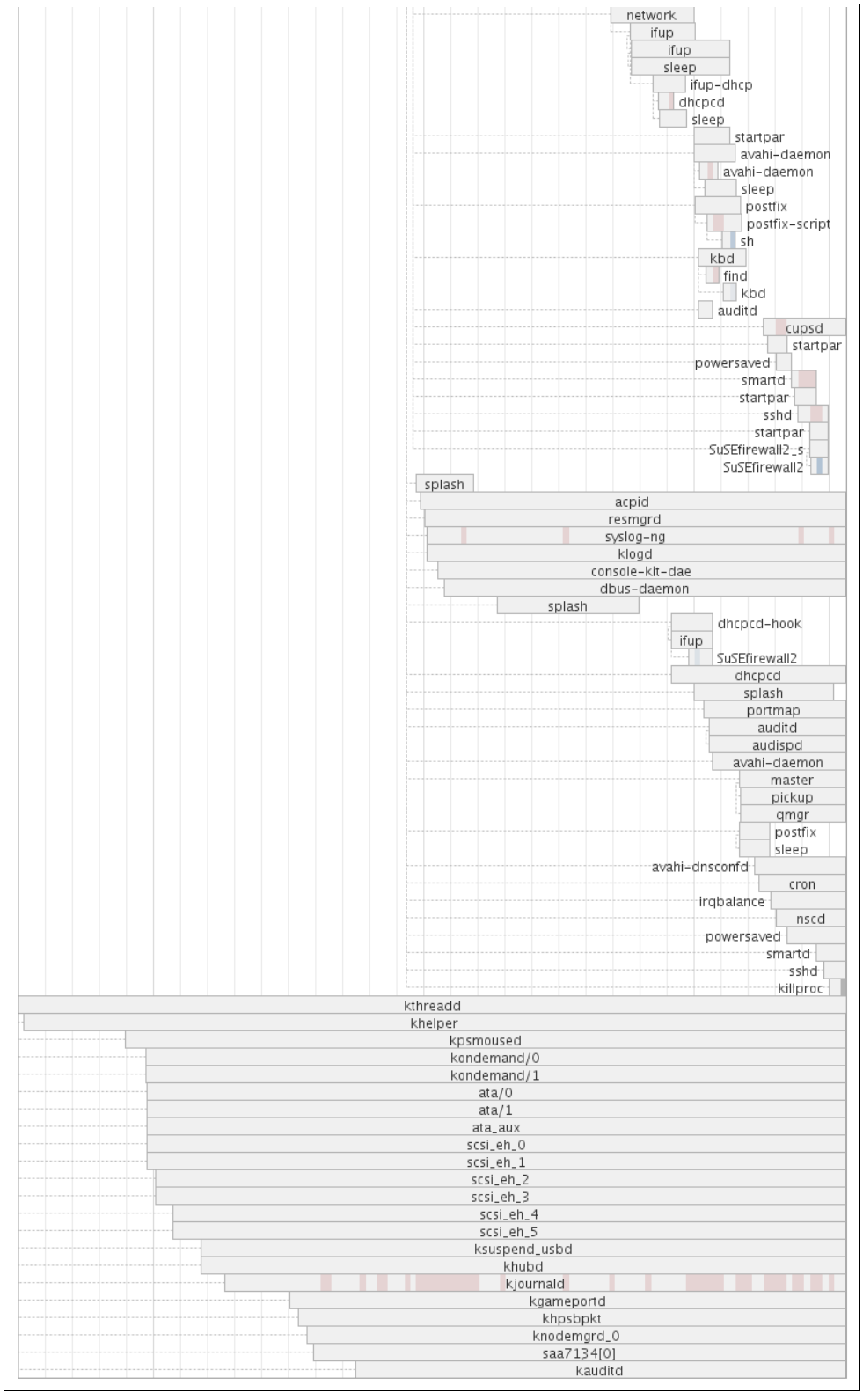


Figure E.6: Boot chart of boot process of openSUSE 10.3 RC1 on System ATH64X2_2.0 with SEA3.5SATA500GB (Part 2)

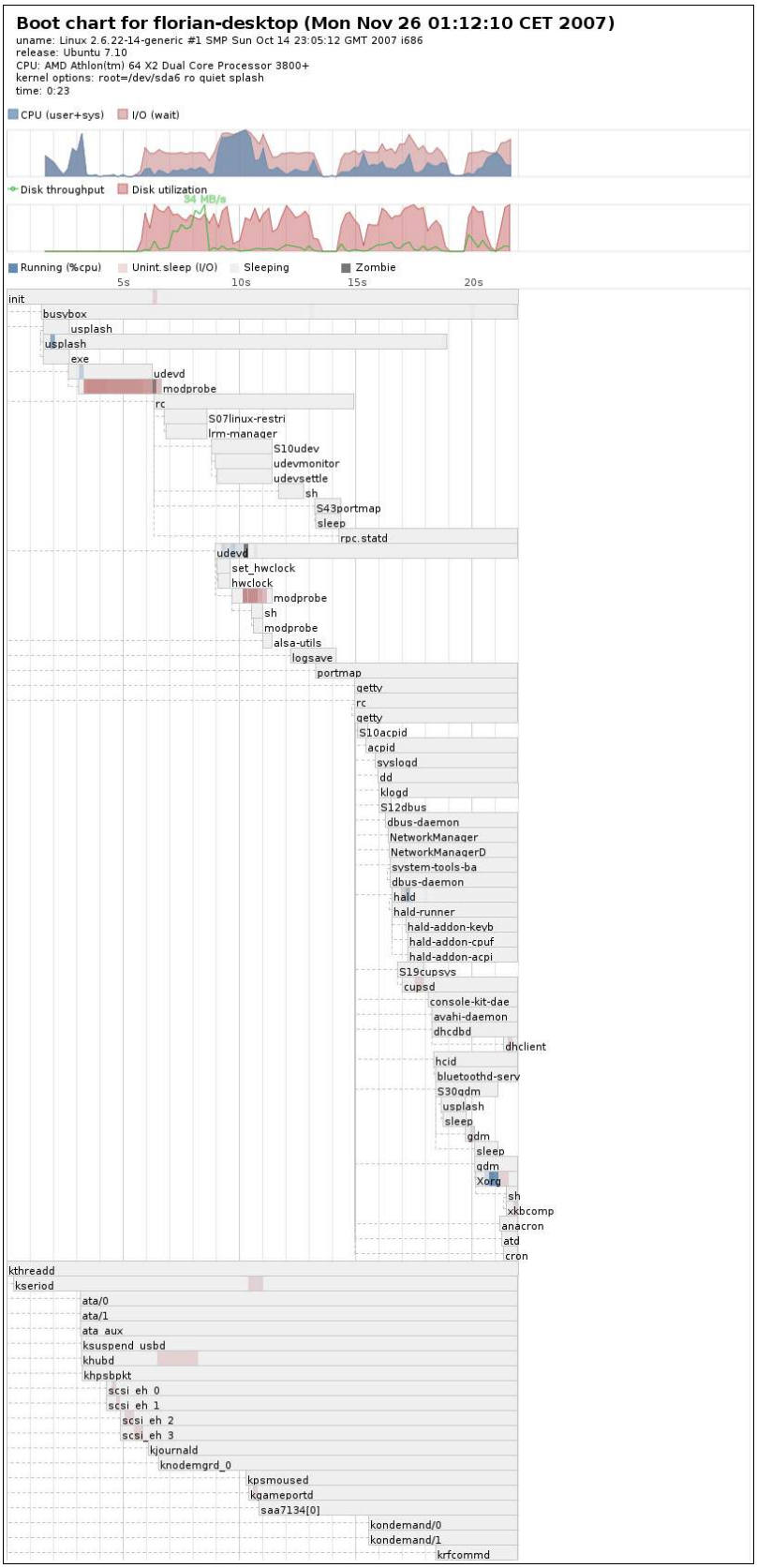


Figure E.7: Boot chart of boot process of Ubuntu 7.10 beta with SEA3.5SATA500GB without use of readahead-list

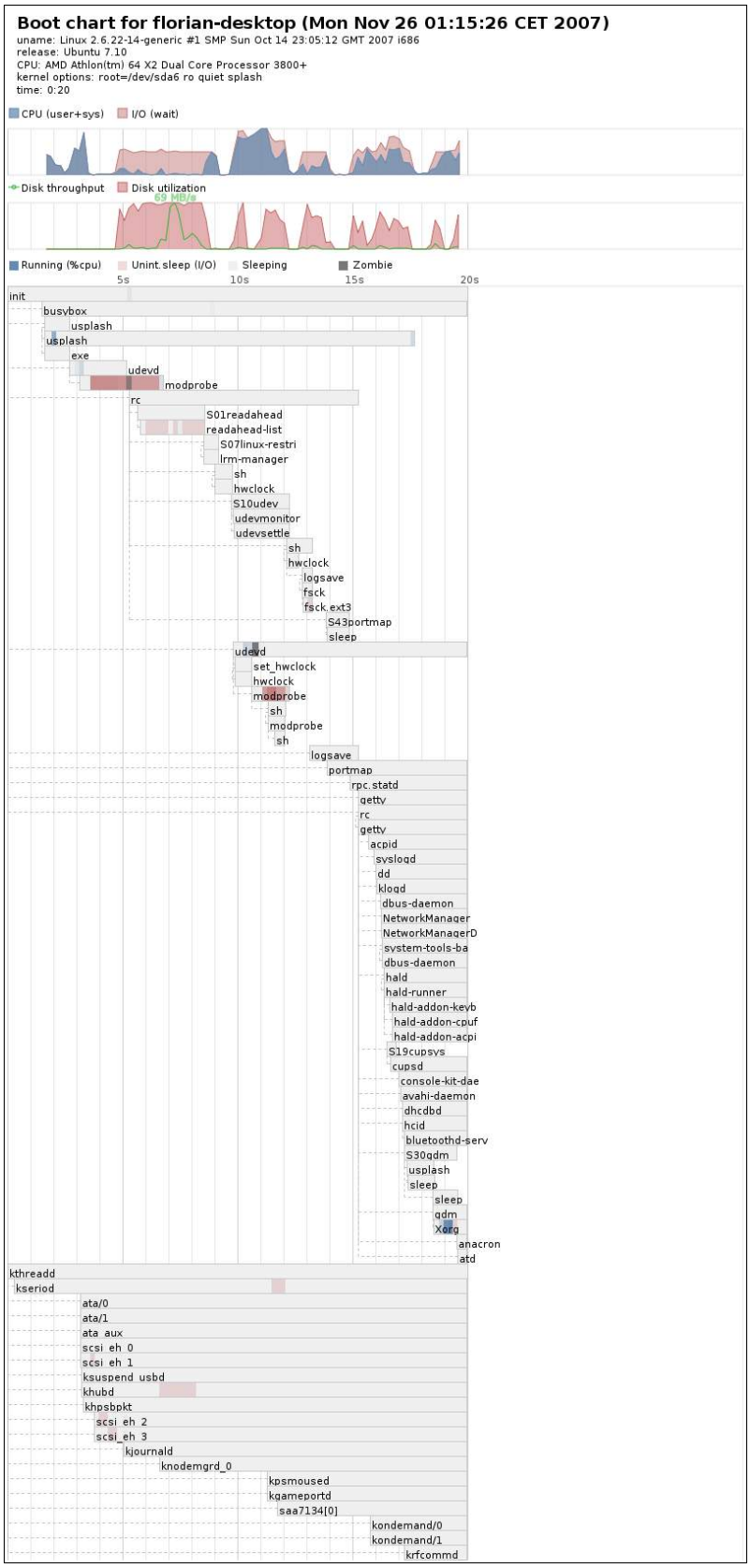


Figure E.8: Boot chart of boot process of Ubuntu 7.10 beta with SEA3.5SATA500GB with use of readahead-list

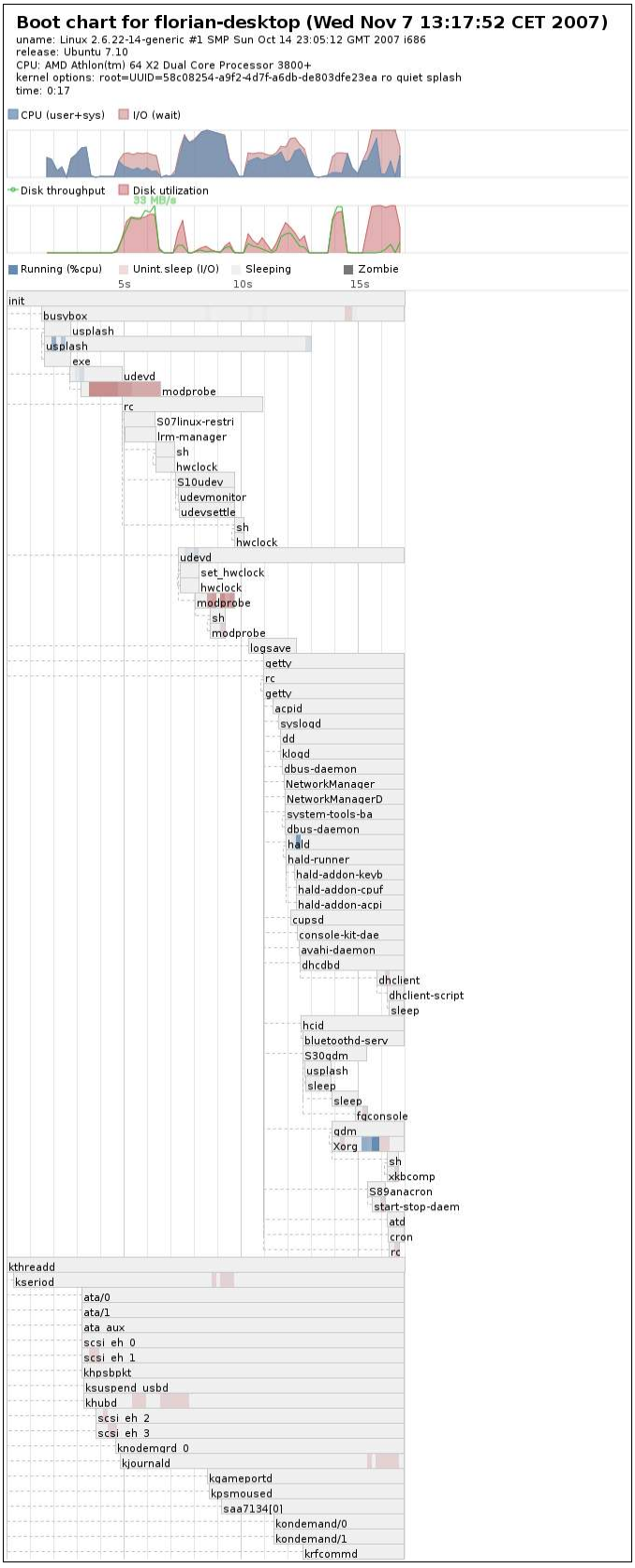


Figure E.9: Boot chart of boot process of Ubuntu 7.10 beta with TRACF4GB without use of readahead-list

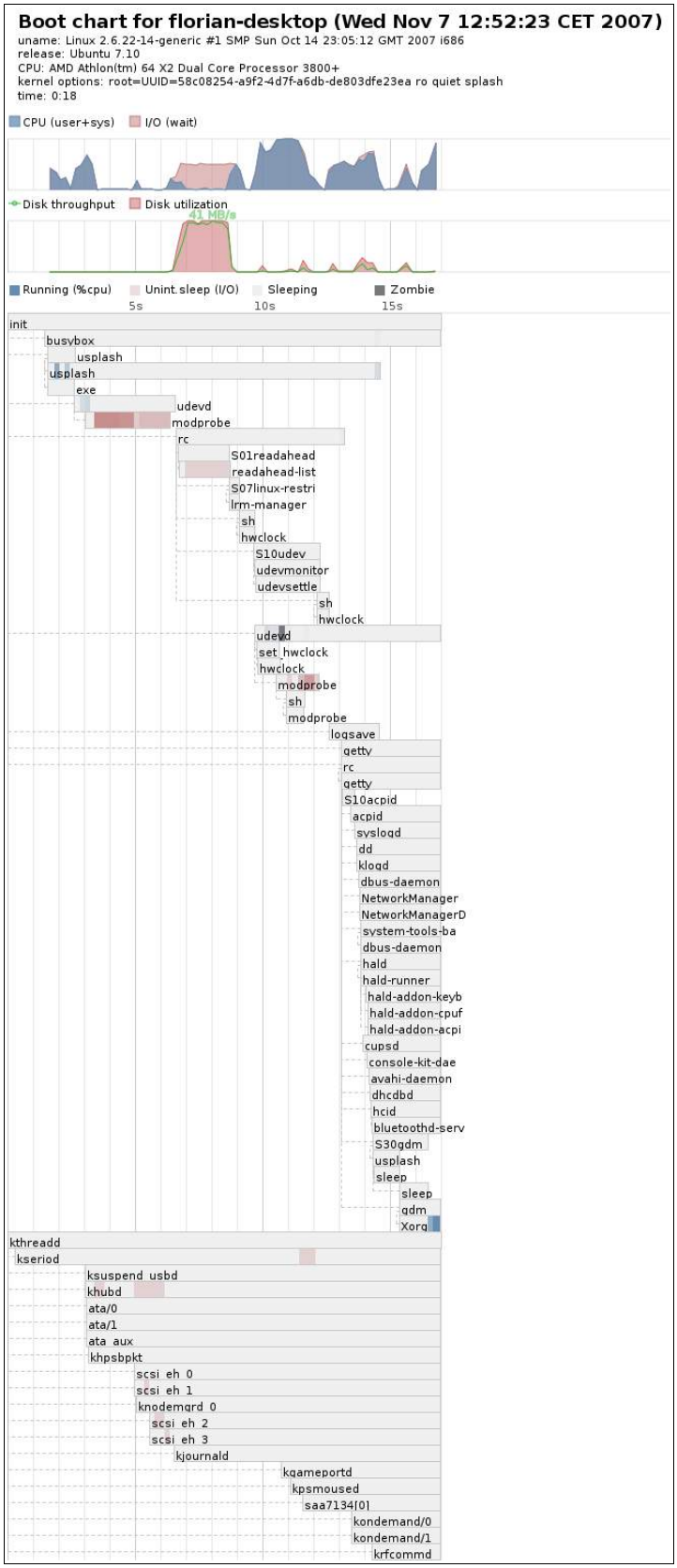


Figure E.10: Boot chart of boot process of Ubuntu 7.10 beta with TRACF4GB after reprofiling readahead-list

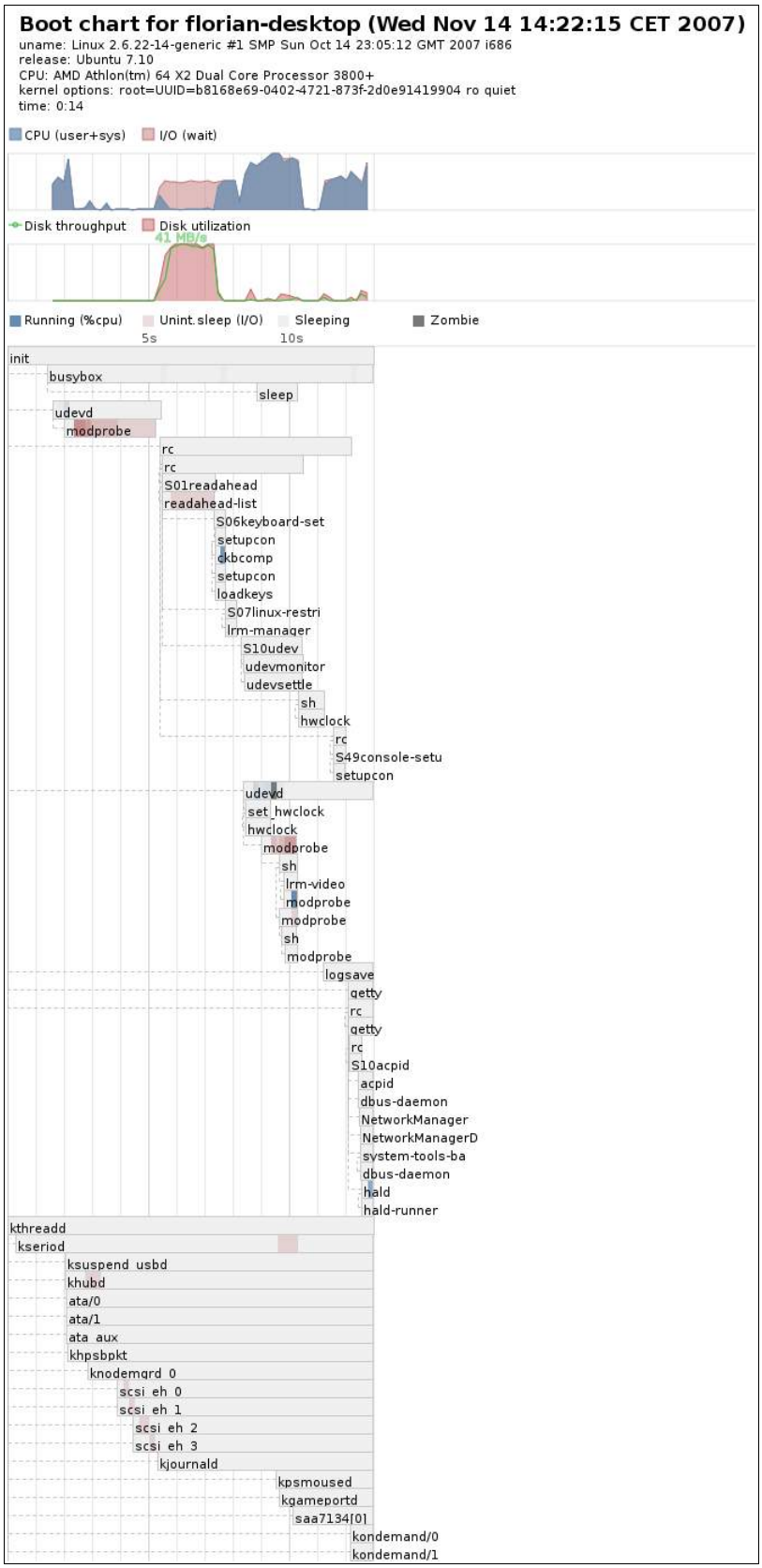


Figure E.11: Boot chart of boot process of the tweaked Ubuntu 7.10 beta with TRACF4GB

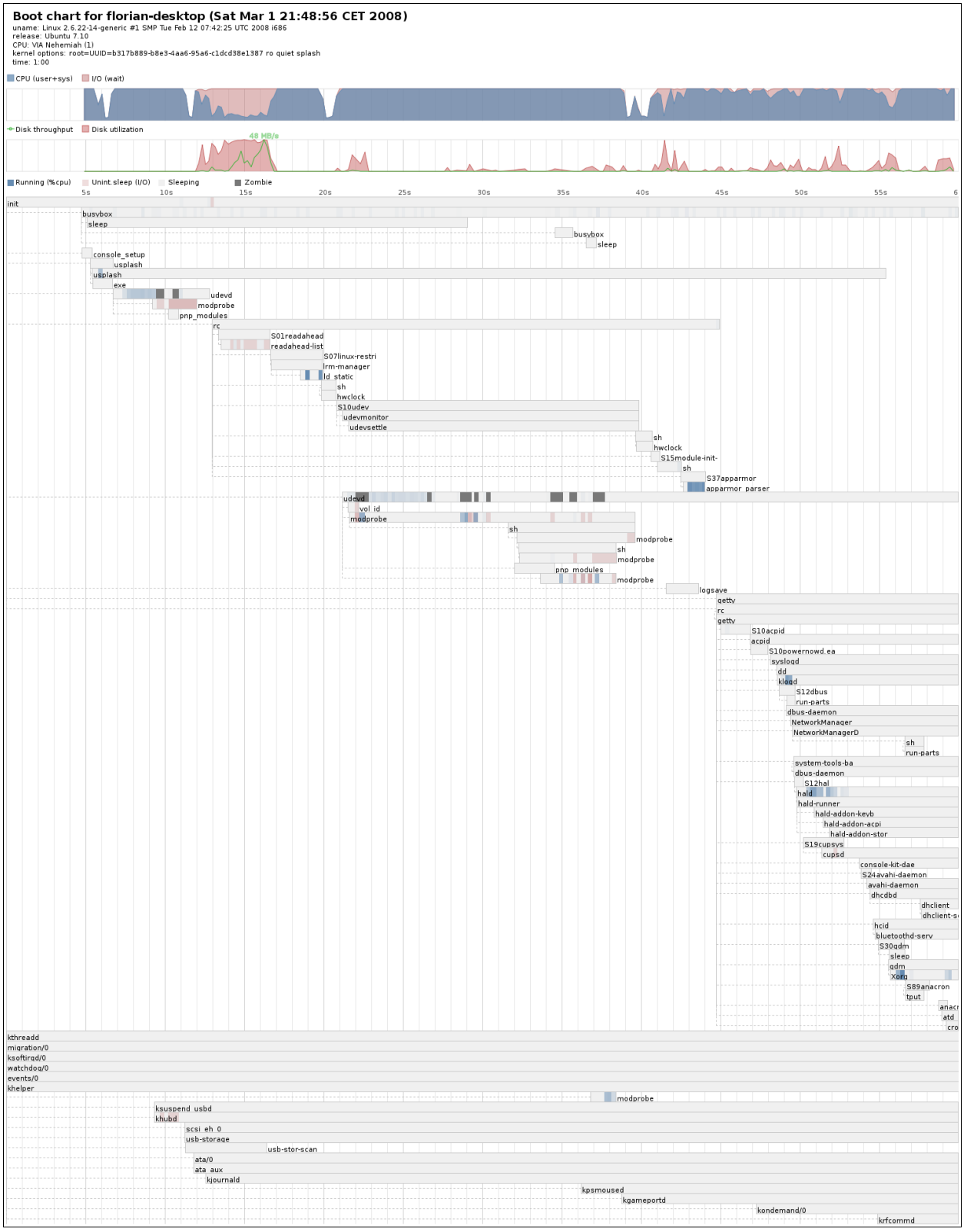


Figure E.12: Boot chart of boot process of Ubuntu 7.10 beta on System EPIA-ML with SEA3.5SATA500GB

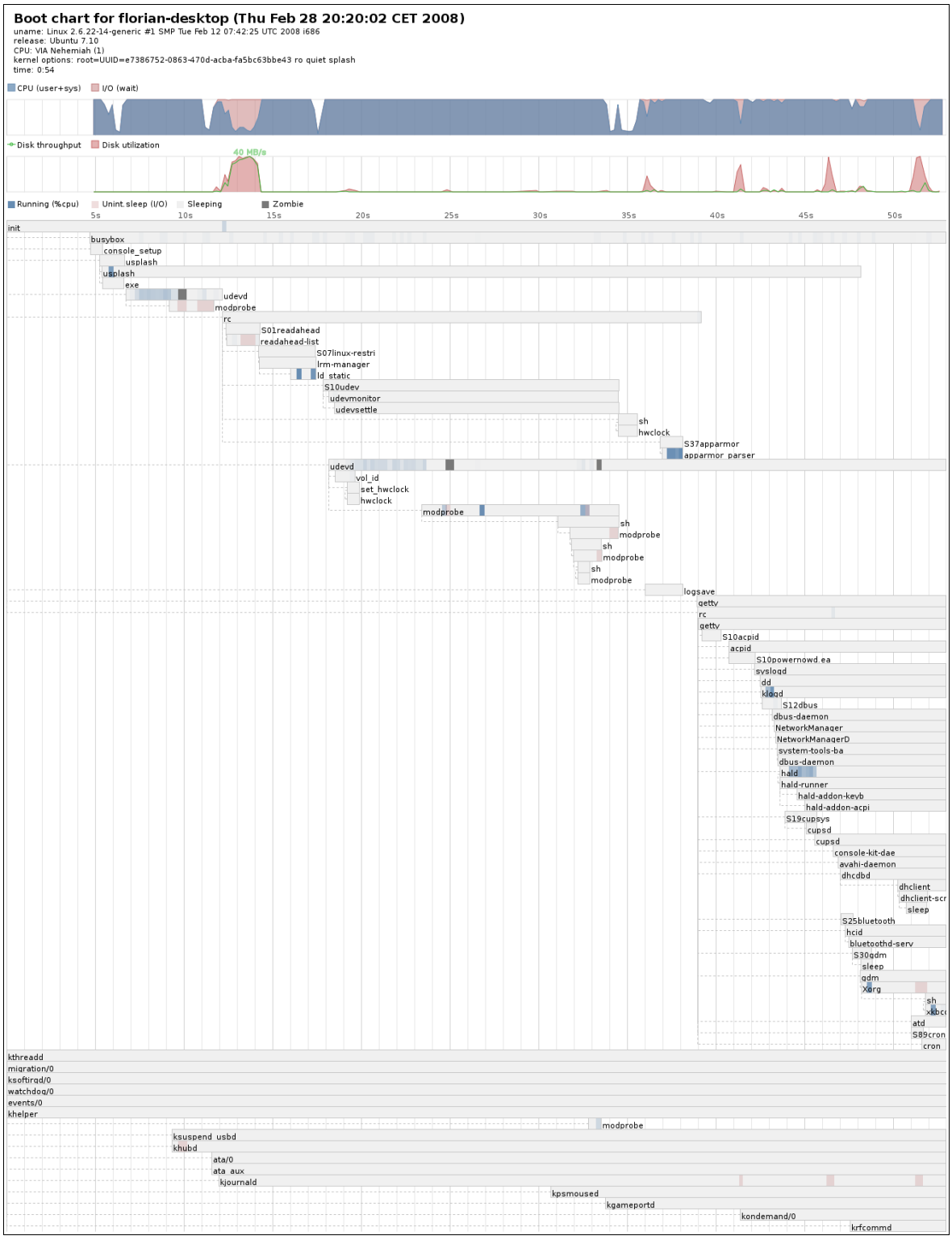


Figure E.13: Boot chart of boot process of Ubuntu 7.10 beta on System EPIA-ML with TRACF4GB

F Additional tables

F.1 Features list of Suspend-To-Disk and Linux

Name	swsusp	TuxOnIce	uswsusp
Available in kernel	2.6.x	patch against 2.6.x	since 2.6.17
Kernel config option	CONFIG_HIBERNATION	CONFIG_TUXONICE	CONFIG_HIBERNATION
Principle author	Rafael Wysocki	Nigel Cunningham	Rafael Wysocki
PM subsystem required	none	none	none
Telling kernel at boot where to save image	resume=/dev/hda#	resume=<file:| swap:>/dev/ [node]<:sector> If file & swap allocators are both compiled in, swap is the default.	
How to activate suspend	echo -n disk > /sys/power/state	echo > /sys/power/tuxonice/do _suspend or echo disk > /sys/power/state if replacing swsusp is enabled.	Userspace program
Telling kernel not to try and resume in case of a problem	noresume	noresume	?
Architecture support	i386, ppc, x86_64, ia64	i386, ppc, x86_64, ia64	i386, ppc, x86_64, ia64
Max. Image size	1/2 memory	See below	1/2 memory
Highmem support	Yes (up to 4GB)	Yes (up to 4GB)	Yes (up to 4GB)
Discontinuous memory support	Yes	Yes	Yes
SMP support	Yes	Yes	Yes
Preemption support	Yes	Yes	Yes
Compression	No	via cryptoapi - LZF recommended	Yes, with libraries
Encryption	No	by writing to dm-crypt partition	With libraries
Suspend-to-swapfile support	No	Yes	Yes?
Suspend-to-multiple swap partitions/files	No	Yes	No
Suspend-to-file support	No	Yes	No
Modular support	No	Yes (post 2.2.9)	No

Initrd support (needed for LVM/dm-crypt)	Yes	Yes	Required
UML support	No	No	No
Wake alarm support	No	Yes	No
Cluster support	No	In progress	No
Suspend-over-NFS support	No	No (also planned ;)	No
Auto swapon when starting to hibernate	No	Yes	No
Easy location of resume= values	No	Yes	No
Reconfigure without rebooting	No	Yes	No
Full mem. accounting, leak detection, failure path testing.	No	Yes	No
Expected compression ratio to avoid freeing too much mem.	No	Yes	No
Scripting support.	No	Yes	No
Keep image mode (For kiosks).	No	Yes	No
Mark resume attempted (sane default if resuming fails).	No	Yes	No
Multithreaded I/O	No	Yes	No
Readahead.	No	Yes	No
Cluster support.	No	In progress	No
Interactive debugging	No	Yes	No
Cancel hibernating via keyboard	No	Yes	Yes
Cancel resuming via keyboard	No	Yes	No
Switch poweroff method while hibernating	No	Yes	No
Checksummed image	No	Yes	No
Unloadable modules when not in use	No	Yes	No
Fuse support	No	Preliminary	No

Table F.1: Features list of Suspend-To-Disk and Linux, source [TOlb]

F.2 Description of the tables TuxOnIce boot

Record	Description
Used memory [MiB]	the memory, which is totally used (get with free)
Free memory [MiB]	the memory, which is not used (get with free)
Buffers [MiB]	the memory, which is used as buffer (get with free)
Cached [MiB]	the memory, which is used as cache (get with free)
Image size uncompressed [MiB]	the size of the hibernation image in uncompressed state (get with debug_info)
Image size compressed [MiB]	the size of the compressed hibernation image (get with debug_info)
Compression in percent	rate of the compression (get with debug_info)
Write speed [MiB/s]	the uncompressed volume of the throughput to the disk by writing the image (get with debug_info)
Read speed [MiB/s]	the uncompressed volume of the throughput from the disk by reading the image (get with debug_info)
Time for read image [s]	calculated time for read the image
Time until TuxOnIce starts [s]	the uptime of the Linux kernel, which is elapsed to the starting point of TuxOnIce
Time [s]	the time, which is get with a stop watch

Table F.2: Description of the tables TuxOnIce boot

G Source code

G.1 Fbtt

The source code and binaries of Fbtt can be downloaded from [FBT08].

G.1.1 fbtt.cpp

```
/* The source code of Fbtt is standing under the */
/* GNU GENERAL PUBLIC LICENSE Version 2 */
/* for more information about the GPLv2 see */
/* http://www.gnu.org/licenses/info/GPLv2.html */

/* principal author: Florian Strunk */

#include<iostream>
#include<fstream>  // for files
#include<string>  // for string

#include <ctime>  //for time_t

#include <sstream>  // for stringstream
#include <list>
#include <boost/algorithm/string.hpp> // split for string

#include <stdint.h> // for uint64_t ???

using namespace std;

const std::string version = "0.12";

std::string getUpTime()
    {
        ifstream file("/proc/uptime");
        std::string buffer;
        getline(file,buffer);
        file.close();
        return buffer;
    }


uint64_t getTSC()
/* reads and prints timestamp counter on i386/amd64/itanium */
/* source code of getTSC written by Parthey */
/* modified by Florian Strunk */

{
  uint64_t tsc;

#ifdef __i386__
  __asm__(
          "cpuid         \n"   /* no out-of-order execution */
          "rdtsc         \n"
          :
          "=A" (tsc)
          :
          : "ebx", "ecx"          /* cpuid scratcht eax, ebx, ecx, edx */
```

```
          );
#endif

#ifdef __x86_64__
  __asm__(
          "cpuid         \n"   /* no out-of-order execution */
          "rdtsc         \n"
          "shl $32,%%rdx \n"    /* x86_64 uses %rax instead of %dx/%ax
*/
          "or  %%rdx,%%rax \n"  /* to store 64 bit function return values
*/
          :
          "=a" (tsc)
          :
          : "ebx", "ecx", "edx", "cc" /* cpuid scratcht eax, ebx, ecx,
edx */
                                      /* shl+or change the flag register
*/
          );
#endif

#ifdef __ia64__
  __asm__ (
          "srlz.i;        \n"   /* no out-of-order execution */
          "mov %0=ar.itc; \n"
          :
          "=r" (tsc)
          );
#endif
  //printf("%llu", (unsigned long long int) tsc);
  return tsc;
}


std::string IntToStrWithZero(int number,int zeros)
{
  stringstream ss;
  // Anzahl Ziffern herausfinden
  int count = 1;
  int tempnumber = number;

  while (tempnumber > 9)
  {
    tempnumber /= 10;
    count++;
  }

  for (int i = count; i<zeros; i++)
  {
    ss << 0 ;
  }
   ss << number;

   return ss.str();
}

void getDateTime(std::string &str_date, std::string &str_time)
{
  time_t now = time(0);

  tm* localtm = localtime(&now);

  std::stringstream sstr_time;
```

```
  std::stringstream sstr_date;

  sstr_time << IntToStrWithZero(localtm->tm_hour,2) << ":" <<
IntToStrWithZero(localtm->tm_min,2) << ":" << IntToStrWithZero(localtm->
tm_sec,2);
  //cout << sstr_time.str() << endl;

  sstr_date << localtm->tm_year+1900 << "-" << IntToStrWithZero(localtm->
tm_mon + 1,2) << "-" << IntToStrWithZero(localtm->tm_mday,2);
  //cout << sstr_date.str() << endl;

  str_date=sstr_date.str();
  str_time=sstr_time.str();
}

void extractUpIdleTime(string upidletime, string &uptime, string
&idletime)
{
  list<string> results;
  boost::split(results,upidletime,boost::is_any_of(" "));
  uptime = results.front();
  results.pop_front();
  idletime = results.front();
}

void WriteLogFile(string date, string time, string uptime, string
idletime, uint64_t tsc)
{
  // if directory /var/log/fbtt not exist then make it

  string path = "/var/log/fbtt/";
  string sysmkdir = "mkdir -p ";
  system((sysmkdir+path).c_str());

  // generate filename with date and time
  string filenamedate = date;
  string filenametime = time;

  string::size_type pos = 0;
  while ((pos = filenametime.find(':',pos)) && (pos!=std::string::npos))
  {
    // String ":"  change with ""
    filenametime.replace(pos,1,"");
  }
  //cout << filenametime << endl;

  ofstream logfile((path+filenamedate+"_"+filenametime+".log").c_str());
  if (!logfile)
  {
    cerr << "File " << (path+filenamedate+"_"+filenametime
+".log").c_str() << " could not be generated" << endl;
  }
  else
  {
  // Content of the file
  // fbtt version: 0.12
  // date: yyyy-mm-dd
  // time: hh:mm:ss
  // boottime: s.ss
  // idletime: s.ss
  // tsc: uint64_t
  logfile << "fbtt version: " << version << endl;
  logfile << "date: " << date << endl;
```

```
  logfile << "time: " << time << endl;
  logfile << "boottime: " << uptime << endl;
  logfile << "idletime: " << idletime << endl;
  logfile << "tsc: " << tsc << endl;
  }
  logfile.close();
}

int main()

{
  string date,time,upidletime, uptime, idletime;
  uint64_t tsc;
  upidletime = getUpTime();
  tsc = getTSC();
  getDateTime(date,time);
  extractUpIdleTime(upidletime,uptime, idletime);
  WriteLogFile(date,time,uptime, idletime,tsc);
  cout << "fbtt version: " << version << endl;
  cout << "date: " << date << endl;
  cout << "time: " << time << endl;
  cout << "boottime: " << uptime << endl;
  cout << "idletime: " << idletime << endl;
  cout << "tsc: " << tsc << endl;
}
```

G.1.2 makefile

```
BINARY=fbtt
CPPFLAGS=-Wall -O3
all: $(BINARY)
install:
      cp $(BINARY) /
uninstall:
      rm /$(BINARY)
clean:
      rm -f $(BINARY) *.o *~
```

Glossary

BIOS	Basic Input Output System
CD	Compact Disk
CF	Compact Flash
CGA	Color Graphics Adapter
COM	Communication Equipment
CPU	Central Processing Unit
DHCP	Dynamic Host Configuration Protocol
DMA	Direct Memory Access
DVD	Digital Versatile Disc
EFI	Extensible Firmware Interface
EISA	Extended Industry Standard Architecture
ELC	Embedded Linux Conference
EPS	Encapsulated Postscript
ext3	third extended filesystem
Fbtt	Florian's Boot Time Tool
FeRAM	Ferroelectric Random Access Memory
GDM	GNOME Display Manager
GiB	Gibibyte (2^{30} Byte)
GNOME	GNU Network Object Model Environment
GNU	GNU's Not Unix
GRUB	GRand Unified Bootloader
HAL	Hardware Abstraction Layer
HDD	Hard Disk Drive
I/O	Input/Output
IA32	Intel Architecture 32-Bit
IDE	Integrated Drive Electronic
KDE	K Desktop Environment
KDM	KDE Display Manager
KiB	Kibibyte (2^{10} Byte)
LAN	Local Area Network
Mac	Macintosh
MiB	Mebibyte (2^{20} Byte)
MRAM	Magneto resistive Random Access Memory

NFS	Network File System
NMI	Not Maskable Interrupt
NRAM	Nano Random Access Memory
OLS	Ottava Linux Symposium
OS	Operating System
PIT	Programmable Interval Timer
PNG	Portable Network Graphics
POST	Power On Self Test
PRAM	Phase-change Random Access Memory
PVR	Personal Video Recorders
PXE	Preboot eXecution Environment
RAID	redundant array of independent disks
RAM	Random Access Memory
ROM	Read Only Memory
RRAM	Resistive Random Access Memory
SATA	Serial Advanced Technology Attachment
SONOS	Semiconductor-Oxide-Nitride-Oxide-Semiconductor
SPD	Serial Presence Detect
SPI	Software in the Public Interest
SPRAM	Spin transfer torque Random Access Memory
SSD	Solid State Disk
SVG	Scalable Vector Graphics
TSC	Time Stamp Counter
UDMA	Ultra-Direct Memory Access
UEFI	Unified Extensible Firmware Interface
UMPC	Ultra Mobile Personal Computer
USB	Universal Serial Bus
XDM	X Display Manager
ZMD	Zen Management Daemon

References

APPA: AppArmor; http://de.opensuse.org/Apparmor [02-Mar-2008]

AXB06: Axboe, Jens; fcache: a remapping boot cache; 2006; http://lkml.org/lkml/2006/5/15/46 [02-Mar-2008]

BOOa: Bootchart; http://bootchart.org/ [02-Mar-2008]

BOOb: Bootchart - Download; http://bootchart.org/download.html [02-Mar-2008]

BUG: HAL fails to initialise when /etc/init.d/rc sets CONCURRENCY=shell; 2007; https://bugs.launchpad.net/ubuntu/+source/hal/+bug/149881 [02-Mar-2008]

BUGa: xf86-video-amd: fails to autoconfigure; 2007; https://bugs.launchpad.net/ubuntu/+source/xserver-xorg-video-amd/+bug/140051 [09-Mar-2008]

BUGb: xserver-xorg.video-amd crashes on Geode LX/Linutop on Xorg start; 2007; http://lists.freedesktop.org/archives/xorg/2007-October/029336.html [09-Mar-2008]

CON06: Conrad Electronic SE; SATA - COMPACTFLASH KONVERTER Best.-Nr. 974571; 2006; http://www.produktinfo.conrad.com/datenblaetter/950000-974999/974571-in-01-de-HINWEIS_SATA-COMPACTFLASH_KONVERTER.pdf [02-Mar-2008]

COR: Payloads - coreboot; http://www.coreboot.org/Payloads [02-Mar-2008]

CPUZ: CPU-Z; http://www.cpuid.com/cpuz.php [09-Mar-2008]

DEBI: Debian -- Spenden an Software in the Public Interest; 2008; http://www.debian.org/donations [02-Mar-2008]

DIST07: Top Ten Distributionen; 2007; http://distrowatch.com/dwres.php?resource=major [02-Mar-2008]

FBT08: Strunk, Florian; Fbtt - Florian's Boot Time Tool; 2008; http://hardtest.florian-strunk.de/showthread.php?tid=53 [09-Mar-2008]

GIA07: Giannaros, Francis; Sneak Peeks at openSUSE 10.3: Greatly Improved Boot Time; 2007; http://news.opensuse.org/2007/08/12/ [02-Mar-2008]

GIT: GIT source for Ubuntu 71.0 kernel with TuxOnIce; cit clone git://kernel.ubuntu.com/nigelc/ubuntu-gutsy+tuxonice.git ubuntu-gutsy +tuxonice.git [27-Feb-2008]

HEI: c't-Systeminfo; http://www.heise.de/ct/ftp/ctsi.shtml [09-Mar-2008]

HEN08: VIA's New Centaur Designed Isaiah CPU Architecture; 2008; http://enthusiast.hardocp.com/article.html?art=MTQ1MCwxLCxoZW50aHVzaWFzdA== [02-Mar-2008]

JDO06: jdong; Improve bootup speed by reprofiling bootup; 2006; http://ubuntuforums.org/showthread.php?t=254263 [02-Mar-2008]

KOF07a: Kofler, Michael; Linux; 8th; Addison-Wesley; 2007; page 750

KOF07b: Kofler, Michael; Linux; 8th; Addison-Wesley; 2007; page 756

LMA05: Linux Magazin; Boot-Beschleuniger im Vergleich; Linux New Meda AG; 11/2005; page 64-69

LMA07a: Linux Magazin; Linux führt bei Embedded-Systemen; Linux New Meda AG; 12/2007; page 750

LMA07b: Linux Magazin; Schnelles Booten mit Upstart, einem Ersatz für das betagte Sys-V-Init; Linux New Meda AG; 02/2007; page 72-77

LMA08: Linux Magazin; Leichtgewichtige Desktopumgebungen im Test; Linux New Meda AG; 03/2008; page 60-63

MCT08: Magazin für Computertechnik (c't); Intels Silverthorne für Embedded Systems; Heise Zeitschriften Verlag GmbH & Co. KG; 06/2008; page 26

MIN04: Minnich, G. Ronald; LinuxBIOS at Four; 2004; http://www.linuxjournal.com/article/7170 [02-Mar-2008]

MOO65: Moore, Gordon; Moore's Law; 1965; http://www.intel.com/technology/mooreslaw/ [02-Mar-2008]

NEC07: NEC Develops World's Fastest SRAM-Compatible MRAM; 2007; http://www.nec.co.jp/press/en/0711/3001.html [02-Mar-2008]

NFS: Linux NFS FAQ; http://nfs.sourceforge.net/ [02-Mar-2008]

OPD07: Opdenacker, Michael; Readahead - Time Travel Techniques; 2007; http://tree.celinuxforum.org/CelfPubWiki/OttawaLinuxSymposium2007 [09-Mar-2008]

PAR06: Parthey, Daniel; Booting Linux Really Fast; 2006;

PAR06a: Parthey, Daniel; Booting Linux Really Fast;2006; page 29

PAR06b: Parthey, Daniel; Booting Linux Really Fast;2006; page 7-10

PAR06c: Parthey, Daniel; Booting Linux Really Fast;2006; page 27

PAR07: Parthey, Daniel, Analyzing real-time behavior of flash memories, 2007

SCH07: Schulz, Hans-Peter; Linux Kompendium - POST Ablauf Vers. 4.53; 2007; http://www.bios-info.de/4p92x846/awpost53.htm [02-Mar-2008]

SCL06: Schenk, Lars; PXE Netzwerk-Boot mit Ubuntu Client & Debian Server; 2006; http://lars-schenk.com/pxe-netzwerk-boot-mit-ubuntu-client-und-debian-server/60 [09-Mar-2008]

SUS07a: Boottime; 2007; http://en.opensuse.org/Boottime#fcache [02-Mar-2008]

SUS07b: Ubuntu and Upstart; 2007; http://en.opensuse.org/Ubuntu_and_Upstart [02-Mar-2008]

TEC08: Intel Silverthorne: 2-GHz-Mobile-CPU mit neuer Architektur; 2008; http://www.tecchannel.de/pc_mobile/news/1745992/ [02-Mar-2008]

THE07: TheeMahn2003; Tweaking Ubuntu Ultimate; ; ubuntusoftware.info/Howto_tweak_ubuntu_ultimate.html [09-Mar-2008]

TOIa: TuxOnIce - Download; http://www.tuxonice.net/downloads/ [09-Mar-2008]

TOIb: Software Suspend - Features; http://www.tuxonice.net/features [09-Mar-2008]

TOIc: Cunningham, Nigel; Read Image with tiny kernel to reduce boot time; http://lists.tuxonice.net/lurker/message/20080106.235327.9b150378.en.html [09-Mar-2008]

TOId: Software Suspend HOWTO - Keep Image Mode; http://www.tuxonice.net/HOWTO.html#toc7.5 [09-Mar-2008]

TOP07: TOP500 Supercomputing Sites; 2007; http://www.top500.org [02-Mar-2008]

UBD07: Diskless Ubuntu Howto; 2007; https://help.ubuntu.com/community/DisklessUbuntuHowto [09-Mar-2008]

UBM07: Ubuntu Mobile; 2007; http://www.ubuntu.com/products/mobile [09-Mar-2008]

UEFI1.1: TCG EFI Platform Specification Version 1.20 Final Revision 1.0; 2006; https://www.trustedcomputinggroup.org/specs/PCClient/TCG_EFI_Platform_1_20_Final.pdf [02-Mar-2008]

UEFI2.1: UEFI Specification Version 2.1; http://www.uefi.org/specs/ [02-Mar-2008]

UPS07: upstart - event-based init daemon; 2007; http://upstart.ubuntu.com/ [02-Mar-2008]

WIKIa: Wikipedia; Keyword: Debian; http://de.wikipedia.org/debian [02-Mar-2008]

WIKIb: Wikipedia; Keyword: Ubuntu; http://de.wikipedia.org/wiki/Ubuntu [02-Mar-2008]

WIKIc: Wikipedia; Keyword: OpenSUSE; http://de.wikipedia.org/wiki/OpenSUSE [02-Mar-2008]

WIKId: Wikipeda; Keyword: Booten; http://de.wikipedia.org/wiki/Booten [02-Mar-2008]

WOO07: Wool, Vitaly; Parallelizing Linux boot on CE Devices; 2007; http://tree.celinuxforum.org/CelfPubWiki/ELCEurope2007Presentations [02-Mar-2008]

YAST: Yast - OpenSuSE; http://de.opensuse.org/YaST [02-Mar-2008]

YOU07: How to open a mac mini; 2007; http://www.youtube.com/watch?v=Q5DdtVHjI5M&feature=related [09-Mar-2008]

Declaration of Authorship

I hereby declare that the whole of this diploma thesis is my own work, except where explicitly stated otherwise in the text or in the references. This work is submitted to Chemnitz University of Technology as a requirement for being awarded a diploma in Applied Computer Science (“Angewandte Informatik”). I declare that it has not been submitted in whole, or in part, for any other degree.

Chemnitz, March 10, 2008

Florian Strunk