Analyzing ChromeOS's Boot Performance

Alexis Emperador
The University of Texas at Austin
aemperador83@yahoo.com

Alison N. Norman
The University of Texas at Austin
ans@cs.utexas.edu

May 10, 2013

1 Abstract

ChromeOS, Google's operating system, advertises a boot performance of 8 seconds. It only exhibits this boot time on specific hardware, which suggests that it is able to skip identifying the hardware during the boot process. ChromiumOS, the open source project version of ChromeOS, requires 35 seconds to boot. We hypothesize that the difference in boot time is due to ChromiumOSs need to identify its hardware. To test our hypothesis, we modify ChromiumOS so that it, too, is hardware-specific. Hypothetically, it will exhibit boot performance similar to ChromeOS after our modifications. We have analyzed the boot performance and outputs of the modified version of ChromiumOS and identified factors associated with decreases or increases in boot time.

2 Introduction

Due to the increasing use of web browsers in recent years, companies have introduced web-based operating systems, which are scaled down versions of their traditional operating system counterparts. One web-based operating system in particular, ChromeOS, has a boot time of 8 seconds [2].

The purpose of this research is to analyze the boot performance of ChromeOS and modify ChromiumOS, the open source version of ChromeOS, to have a similar boot performance. Since the ChromeOS boot process is hypothesized to be hardware specific, we modify the ChromiumOS kernel to remove unnecessary checks. We measure the time spent in each stage of the boot process after each modification to identify where time is being spent.

3 Related Work

In this section, we describe a work related to our research about a similar operating system's boot performance.

ChromeOS boot performance is hypothesized to be due to use on a specific architecture. Therefore, if an operating system with similar specifications is hardware specific, it should also exhibit a boot performance similar to ChromeOS.

Ulf Magnusson created a semi-automatic method for generating a *Linux kernel of minimal size* for a given platform. Magnusson's method was tested with Awesome-O, a Linux-based, Web-oriented operating system much like ChromeOS. Magnusson showed that the boot performance for Awesome-O could be increased by minimizing the Linux kernel by removing unnecessary features [4].

4 Background

In this section, we define terms and concepts that are needed to understand ChromeOS and ChromiumOS.

According to Google, *ChromeOS*, Google's operating system, is designed to get the user to the web browser faster. Google claims that there is no reason that a user should have to wait 45 seconds, the average boot time for modern operating systems, to get to the web browser. The average user spends 90 percent of their time on the internet in a web browser application [3]. In ChromeOS, the operating system is the web browser.

Google ships ChromeOS on *Chromebooks* which are laptops with specific hardware and configuration. Samsung released the first Chromebooks in June 2011 [2]. ChromeOS has a boot time of 8 seconds on the Chromebooks.

ChromiumOS is the open source project version of ChromeOS maintained by Google [7]. ChromiumOS and ChromeOS fundamentally share the same code base, but ChromeOS has some additional firmware features and comes with binary packages that are not released with the ChromiumOS project [7].

ChromiumOS has different kernels built in for different architectures. For each architecture, ChromiumOS uses a combination of kernels based on the flavour used. For example, an i386 architecture will use a combination of the pinetrail and menlow flavours. Figure 1 shows a flow chart of the kernel structure for ChromiumOS.

Web-based operating systems are a new class of operating systems that focus on web browsing and web applications [10]. Web-based operating systems are designed to keep the user in the browser longer making use of external network resources.

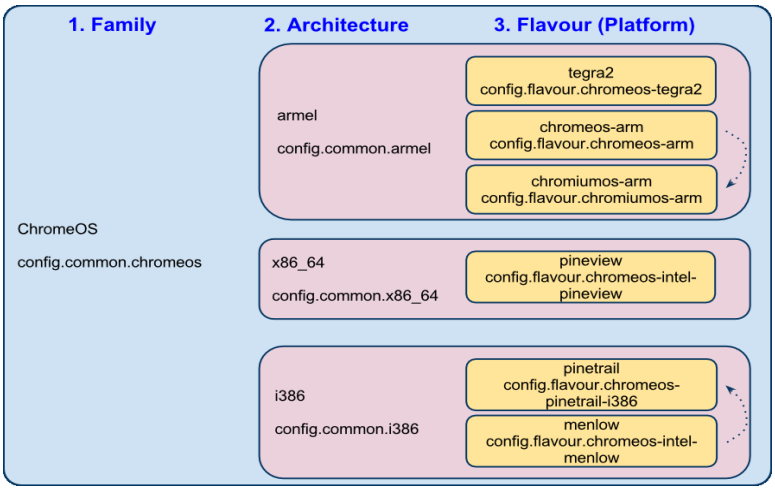

Figure 1: This figure shows the ChromiumOS kernel breakdown and depicts the ChromiumOS kernel with architectures and flavours [5].

5 Problem

We want to identify the factors that contribute to the boot performance of ChromeOS. We know that ChromeOS is shipped on a specific hardware in the form of Chromebooks which we hypothesize contributes to the boot performance. ChromeOS and ChromiumOS boot processes have three stages: 1) BIOS startup and kernel load, 2) start the kernel and execute, and 3) user process startup [1]. Most of the boot time is spent in stage 2, so our goal is to modify ChromiumOS kernel to reduce boot time.

We test our hypothesis using a 32-bit Toshiba NB205 with 2GB DDR2 RAM[1] and an Intel Atom processor.

ChromiumOS version 27.0.1442.0 was used and modified to test our hypothesis. Development and test images with no verified boot enabled were used.

Following the instructions on the *Quick Start Guide* [9], we downloaded the source code and created a chroot[2] on our system for source code. Inside the chroot is the source code to build ChromiumOS and existing scripts to use on the operating system. The source code for the kernel is also located inside the chroot.

Measurements of the boot startup time were recorded using the `bootperf` script provided by the Chromium Project [6]. The `bootperf` script measures ten iterations of the boot cycle.

5.1 Procedure

In this section, we describe the procedure for modifying the kernel and measuring the boot time once modified.

An unmodified version of ChromiumOS was obtained to measure the initial boot performance of ChromiumOS. We will refer to this as the *vanilla build*.

We created an x86-generic build with the flags `noenable_rootfs_verification`, `dev`, and `autotest`. The `noenable_rootfs_verification` flag disables ChromiumOS's check for integrity during the boot process. The `dev` flag creates a development build and the `autotest` flag enables the operating system to be controlled remotely using the test scripts inside the chroot.

After the build was created, we modified the kernel for the build. This was done using the kernel configuration scripts located inside the build [5]. The scripts allow for modification of the three kernel configurations at once by running `make menuconfig` on each kernel. The modifications, see Section 5.2, were made to each kernel configuration. After the modifications were complete, we compiled the kernel so the build used the new modified kernel.

After the build with the modified kernel was ready, we loaded ChromiumOS on a flash drive then boot our machine from the flash drive. We then used the bootperf script from the chroot to remotely run tests on the machine running ChromiumOS.

5.2 Modifying the Kernel

Since ChromiumOS has three stages to its boot process, two of which are focused in the kernel, we have modified the kernel to remove unnecessary checks based on our hardware. Since we tested our hypothesis on a Toshiba netbook, there was no need for certain checks related other architectures.

We started by removing checks for the following in all three ChromiumOS kernel versions:

- Acer Aspire One temperature and fan driver
- Dell WMI extras
- HP WMI extras
- ChromeOS laptop
- ChromeOS LED keyboard
- Thermal Management driver for Intel menlow platform
- ChromeOS specific ACPI extensions

[1] The size of RAM is negligible since we are not testing user space. ChromiumOS limits the number of tabs a user can have open in their browser to the size of RAM [8].

[2] The chroot stands for chrome root. It is where the source code is located for ChromiumOS [8].

- Unnecessary hardware monitoring support
- Allow booting SMP kernel on uniprocessor systems

We were able to remove the checks listed above because these checks were all related to items that were not necessary for our testing architecture. After removing these checks, there was a slight improvement in boot time. The time spent in the kernel was also reduced.

Next, we observe that the kernel does not need to prompt for development and/or incomplete code/drivers because ChromiumOS downloads a fresh copy of ChromiumOS if it detects errors in the code[3]. Additionally, the kernel does not need support for paging of anonymous memory since ChromiumOS also does not support swap memory [8]. Therefore, the kernel does not need to check to see if it is supported since we know it is not supported. Additionally, ChromiumOS does not need to enable VM event counters for /proc/vmstat since we will not be using VMs. Therefore, further checks removed were from the General Setup of each version of the kernel. These items are:

- Prompt for development and/or incomplete code/drivers,
- Support for paging of anonymous memory, and
- Enable VM event counters for /proc/vmstat.

After removing these checks, the boot time decreased by nearly ten seconds.

6 Results and Analysis

After removing the hardware checks, ChromiumOS boot time was reduced from 35 seconds to 26.2 seconds. The boot process spent an overall average of less time in the kernel stage which caused the reduction in boot time.

As shown in Table 1, the initial boot results for 10 cycles for ChromiumOS have a significant amount of time spent in the kernel stages of the boot process. In Table 2, we see that the boot results for 10 cycles for the modified system have significantly less time spent in the kernel stages. Thus, we have shown that the boot time can be significantly reduced by removing unnecessary checks from the kernel.

In comparison to ChromeOS, the modified version of ChromiumOS takes 18.2 seconds longer to boot. This is an improvement from the vanilla version of ChromiumOS which took 27 seconds longer to boot than ChromeOS.

Table 3 shows the time each operating system spends in each stage of the boot process. The modified version of ChromiumOS is significantly better in comparison to ChromeOS than the vanilla version of ChromiumOS.

Event	Time(milliseconds)	s%	dt
startup	2992	3%	+2992
startup_done	9769	16%	+6777
x_started	23089	8%	+13320

Table 1: ChromiumOS vanilla boot results for 10 cycles. Time is the average total time in milliseconds. s% is the sample standard deviation of the total time as a percentage of the average. dt is the average delta change in time between current and previous stages.

[3]Since we disabled verified boot, the download does not replace the kernel.

Event	Time(milliseconds)	s%	dt
startup	3560	2%	+3560
startup_done	9767	7%	+6207
x_started	18246	2%	+8479

Table 2: ChromiumOS with Kernel Modified to remove hardware checks results for 10 cycles. Time is the average total time in milliseconds. s% is the sample standard deviation of the total time as a percentage of the average. dt is the average delta change in time between current and previous stages.

Event	ChromiumOS	ChromiumOS Modified	ChromeOS
Time until login	6.7 sec	5.3 sec	4.6 sec
Start Kernel	13.3 sec	9.7 sec	1.3 sec
BIOS startup/kernel load	6.7 sec	5.6	1.5 sec

Table 3: Table 3. Comparison of ChromiumOS, ChromiumOS Modified, and ChromeOS. Time until login is the average time it takes to get to a login screen after kernel execution. Start Kernel is the average time spent in kernel execution stage. BIOS startup/kernel is average time spent loading BIOS/kernel.

7 Conclusion and Future Work

As shown in Section 6, the modified version of ChromiumOS has a boot time of 26.2 seconds. The modifications of the kernel reduced the boot time from 35 seconds to 26.2 seconds. The results demonstrate the effectiveness of making ChromiumOS hardware specific.

The decrease in the time spent in the kernel following the modifications show the benefit of making the kernel hardware specific. Making the kernel hardware specific eliminates the unnecessary checks for hardware the architecture does not have or may not need.

The modified version of ChromiumOS did not reach a boot time comparable to ChromeOS. However with further experimentation, we believe that a boot time comparable to ChromeOS is possible.

As already mentioned, we modified the kernel to remove checks for specific hardware in the kernel. Further modifications and removals of checks for hardware in the kernel would be an option for future research. Improvement could also be obtained by modifying the kernel compiler to optimize kernel for a specific architecture.

Finally, investigating the user stage of the boot process and identifying factors contributing to overhead may also reduce boot time further since the user stage accounts for approximately 1.68 seconds of the difference between ChromiumOS and ChromeOS boot time.

8 Acknowledgements

This research was supported by an undergraduate research grant from the National Science Foundation.

References

[1] Richard Barnette. Chromeos boot process. Technical report, 2011.

[2] chrome. Samsung chromebook. Technical report, 2013.

[3] Google Chrome. Chromium os fast boot. Technical report, 2013.

[4] Ulf Magnusson. *A Linux-based, Web-oriented operating system designed to boot quickly.* PhD thesis, Institutionen for datavetenskap, 2011.

[5] The Chromium Projects. Kernel configuration. Technical report, 2013.

[6] The Chromium Projects. Tools for measuring boot time performance. Technical report, 2013.

[7] The Chromium Projects. Chromium os. Technical report, 2013.

[8] The Chromium Projects. Chromium os developer guide. Technical report, 2013.

[9] The Chromium Projects. Quick start guide. Technical report, 2013.

[10] Alex Wright. Ready for a web os? *Technology*, December 2009.