TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies
Department of Software Science

Herman Õunapuu 142797

MANAGING INTEL CPU AND GPU PERFORMANCE ON LINUX BASED OPERATING SYSTEMS

Bachelor's thesis

Supervisor: Ago Luberg
MSc

Tallinn 2018

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Herman Õunapuu 142797

INTELI CPU JA GPU JÕUDLUSE HALDAMINE LINUXI-PÕHISTEL OPERATSIOONISÜSTEEMIDEL

Bakalaureusetöö

Juhendaja: Ago Luberg
MSc

Tallinn 2018

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Herman Õunapuu

21.05.2018

Abstract

This thesis aims to solve Intel CPU and GPU related issues on Linux, such as overheating and a short battery life. To solve the issues, two solutions were created, consisting of a service and a client to control the service. Testing sections of this thesis aim to measure the impact of the solutions created.

GPU manager testing includes 2D, 3D workload testing and video playback tests with 1080p and 4K video samples. The analysis of the results found that by limiting the GPU performance great improvements in power savings and efficiency can be achieved.

CPU manager testing consists of stress testing and timed Linux kernel build testing over various configurations, including tests with no CPU manager, Linux CPU manager and Linux Thermal Daemon. Thermal throttling results and analysis demonstrated that Linux CPU manager did limit the CPU temperatures and was more performant when compared to Linux Thermal Daemon with the same limits, confirming the findings of previous work on optimal thermal throttling methods. Further investigation into Linux Thermal Daemon revealed the possible issue and quick tests with the improved configuration did show an improvement when compared to the stock Linux Thermal Daemon configuration.

Comparison between different governors offered by Linux CPU manager revealed that by limiting the CPU performance the power usage and CPU temperatures can be greatly reduced. Furthermore, the CPU efficiency can also be noticeably improved and overall CPU energy use lowered, which translates to an improvement in battery life.

This thesis is written in English and is 64 pages long, including 4 chapters, 45 figures and 6 tables.

Annotatsioon

Inteli CPU ja GPU jõudluse haldamine Linuxi-põhistel operatsioonisüsteemidel

Antud lõputöö käsitleb Inteli protsessorite ning graafikalahendustega seotud ülekuumenemist ning kõrget energiakulu Linuxi-põhistel operatsioonisüsteemidel. Töö käigus loodi kaks eraldiseisvat programmi, mis lubavad kasutajal hallata nii CPU kui GPU jõudlust. Lahendused pakuvad erinevaid jõudlustasemeid ning sisaldavad endas ülekuumenemisvastast komponenti, mis piirab ennetavalt CPU või GPU jõudlust. Lahenduste loomisele järgneb nende testimine, mõõtmaks eesmärkide saavutamist.

GPU lahenduse testid hõlmavad endas 2D ning 3D koormustesti, lisaks testiti ka lahenduse mõju 1080p ning 4K resolutsiooniga videofailide taasesitusel. Testimisel selgus, et GPU lahenduse kasutamine võimaldab märgatavat energiakulu ning CPU kiibi temperatuuri vähendamist ning efektiivsuse tõstmist jõudluse arvelt.

CPU lahenduse testimiseks kasutati lihtsat koormustesti ning ajavõtmisega Linuxi kerneli ehitamist. Analüüsides temperatuurimuutusi, leiti, et CPU lahendus suudab temperatuure piirata ning on sama temperatuurilimiidi juures parema jõudlusega, kui alternatiivne lahendus *Linux Thermal Daemon*, kinnitades mitmete eelnevate teadustööde leide CPU optimaalse kiiruse piiramise viisi osas. Võrdlus CPU lahenduse erinevate jõudlusvalikute osas näitas, et CPU kiiruse piiramisega on võimalik energiakulu ning temperatuure märgatavalt vähendada. Samuti oli piiramisega võimalik suurendada CPU efektiivsust ning vähendada üldist energiakulu, tänu millele on võimalik pikendada aku eluiga.

Uurides *Linux Thermal Daemon*'i ebastabiilset ning volatiilset käitumist, õnnestus autoril leida võimalik probleemi põhjus. Peale ühe võimaliku lahenduse testimist sai põhjus kinnitust ning muudetud konfiguratsiooniga *Linux Thermal Daemon*'i käitumine oli stabiilsem ja CPU keskmine taktsagedus suurem.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 64 leheküljel, 4 peatükki, 45 joonist, 6 tabelit.

List of abbreviations and terms

CPU	*Central Processing Unit*
GPU	*Graphics Processing Unit*
TDP	*Thermal Design Power*
PC	*Personal Computer*
GUI	*Graphical User Interface*
IPC	*Inter-process communication*
USB	*Universal Serial Bus*
UEFI	*Unified Extensible Firmware Interface*
RAM	*Random Access Memory.*
TIM	*Thermal Interface Material*
OS	*Operating System*
API	*Application Programming Interface*
RAPL	*Running Average Power Limit*
CLI	*Command-line Interface*
LCM	*Linux CPU Manager.* Abbreviation of the CPU solution created as part of this thesis.
LTD	*Linux Thermal Daemon.* Abbreviation of the alternative CPU throttling solution.

Table of contents

List of figures

List of tables

1 Introduction

CPU and GPU temperatures and various heat dissipation methods have been under the attention of researchers, chip makers and device manufacturers for decades. Work on features, such as dynamic frequency and voltage scaling, on-chip thermal protection abilities, turbo mode and increased monitoring, has greatly improved performance, efficiency, and reliability. However, badly designed devices with insufficient cooling capabilities can still get too warm to touch and even overheat and cause loss of work for the user because the on-chip thermal protection features are only used when the temperatures get very close to the critical temperature trip point.

A lot of work has gone into software solutions that try to pre-emptively throttle the CPU in a way that least affects the performance. These include improved CPU schedulers, dynamic thermal management modules in the kernel and user space tools that allow the user to change the performance levels of the CPU and GPU. However, some solutions are either difficult to implement and maintain, don't function optimally or don't exist on Linux based operating systems at all, leaving the user with little control over the performance and thermal management, causing issues like overheating and a short battery life.

This thesis aims to solve the problems associated with overheating and short battery life by creating two user space programs that allow for finer control over the CPU and GPU performance on Intel CPU-s and GPU-s. The CPU solution must be able to provide different levels of performance that can be chosen to improve CPU efficiency, battery life and provide good pre-emptive throttling to prevent overheating. The GPU solution must offer multiple different operating modes in order to control the GPU related power usage, improve GPU efficiency and to prevent GPU induced overheating.

The first half of the thesis focuses on the integrated graphics solution found on most Intel CPU-s, going over the issues and current power saving methods, and proposes a solution for solving the issues with high power usage and overheating. Implementation details are followed by testing, test results and analysis to measure the effectiveness of the solution.

The second half of the thesis discusses the CPU related problems in detail, brings up current methods and solutions for controlling the CPU performance and presents the proposed solution for managing overheating and battery life issues. As with the GPU solution, the CPU solution is followed by testing, test results and analysis, which gives an overview of the behaviour and performance of the proposed solution.

2 Linux GPU manager

The GPU power and performance management portion of this thesis gives an overview of the Intel GPU and GPU related issues. This is followed by the solution implementation details, testing the solution and analysis of the results.

2.1 Introduction

Most modern Intel CPU-s ship with integrated graphics known as Intel HD, UHD or Iris Pro graphics. According to Firefox Hardware Report, as of April 2018 about 89% of Firefox users have an Intel CPU and 66% use the Intel integrated graphics [1]. The difference between those two statistics is easily explained by the presence of dedicated GPU-s by AMD and Nvidia used in some desktops and laptops which account for about 28% of Firefox users.

Finding the true market share of Linux based operating systems is not simple due to the different statistic provided by various sources. For example, StatCounter reports Linux usage as of April 2018 at 1.66% [2], as shown on Figure 1.

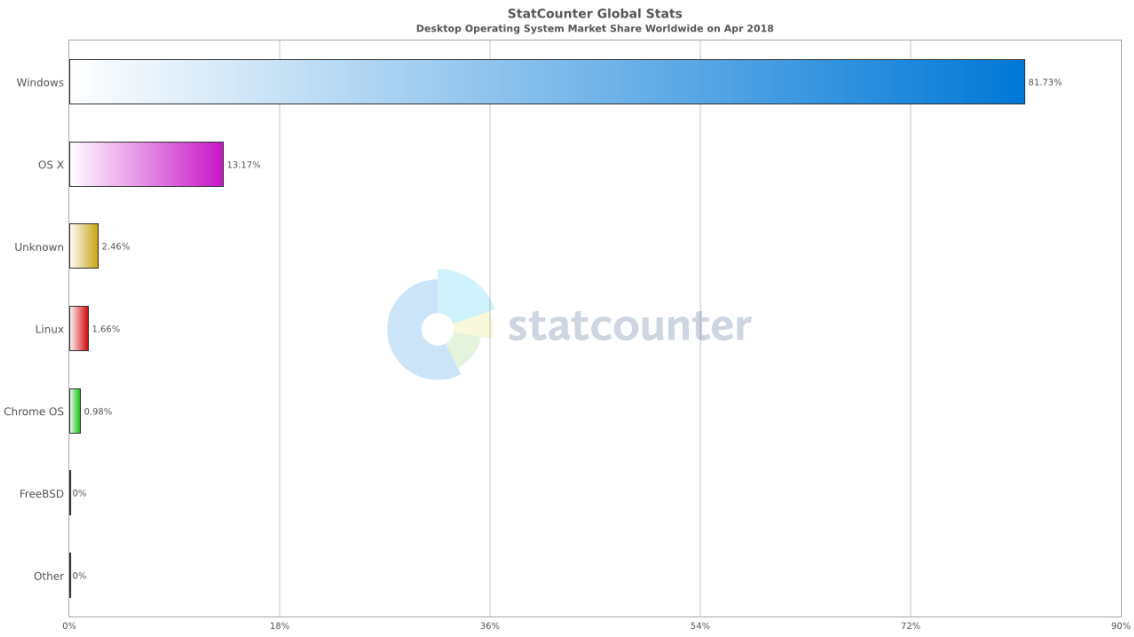

Figure 1. Desktop operating system marketshare according to StatCounter as of April 2018.

Other market share statistics providers, such as NetMarketshare, show that Linux market share is closer to 2%, at 1.93% [3]. Firefox hardware report puts the market share of Linux to 2.73% as of April 2018, possibly due to Firefox being the default browser in most Linux distributions.

Even though the general market share of Linux users is low, a sizeable part of those users could benefit from an Intel GPU manager solution that provided the user with more control over the performance and power usage.

2.2 Issues

Intel integrated graphics solutions can cause some issues during normal usage, such as overheating and high power usage. This section focuses on the GPU related issues and gives a better overview of why these problems are an issue and how to potentially solve them.

2.2.1 High power usage

Intel GPU-s are mostly intended for providing basic support for desktop graphics and graphically less intensive workloads, like office work, browsing the internet and watching movies. 3D workloads, such as video games, generally have poor performance compared to GPU-s by AMD and Nvidia.

Popular browsers, such as Firefox and Chromium, make use of GPU acceleration and are continuing utilising the GPU even further. One example of this is the upcoming Firefox component called WebRender which accelerates the rendering of web pages by offloading some of the work to the GPU [4]. With the increasing emphasis on making use of the GPU, its power usage is also increased, leading to higher temperatures and lower battery life. A decrease in battery life is disadvantageous in situations where image quality and frame rate are not as important as battery life.

Table 1 illustrates the general power usage figures that can be seen on an Intel HD 4000 GPU found on the Intel i7-3820QM under a heavy 3D workload. The chosen workload was a short run of FurMark with `turbostat` running in the background with a polling interval of 5 seconds.

Table 1. GPU power usage while idling and under load.

Clock speed (MHz)	State (idle, load)	Average power usage (W)
350	idle	0.85
350	load	4.80
1250	idle	1.10
1250	load	20.80

From the power usages reported it is clear that the GPU contributes noticeably to the system overall power usage. According to specifications by Intel the CPU TDP (thermal design power) is rated at 45W [5] and its maximum power usage has been observed to reach this with a heavy load. This means that the GPU can contribute 10-45% to the overall CPU package power usage when under load.

2.2.2 Lack of power management features on Linux

Intel HD Graphics Control Panel is a Windows program that allows the user to control various GPU related settings, including power usage modes. Figure 2 shows the three power related options offered by the control panel: maximum battery life, balanced and maximum performance. A power mode can be set for when the PC (personal computer) is running on battery or when plugged in.

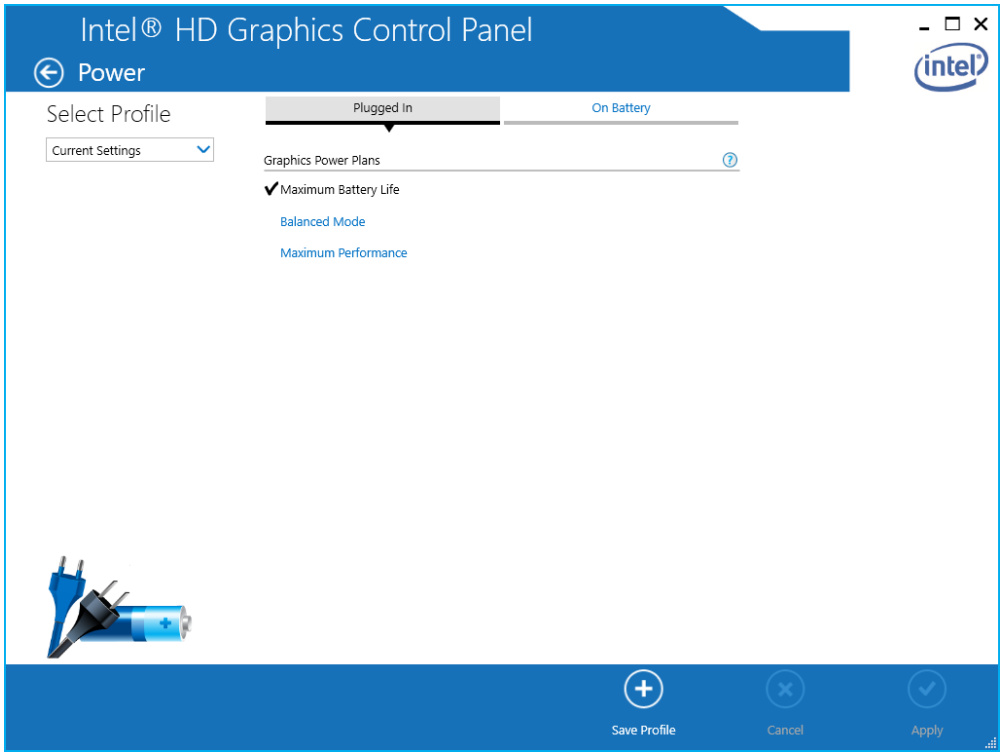

Figure 2. Intel HD Graphics Control Panel power usage controls.

To find out what the power usage and clock speed ranges are for the different modes, two programs are used. OCCT is used for monitoring the GPU clock speed, temperature and for generating a heavy 3D workload. OpenHardwareMonitor, a free open source program for monitoring system sensors, is used to measure the GPU power usage in watts. Results of the quick testing are shown on Table 2. It is clear that running the GPU at a lower clock speed will have a huge positive impact on the power usage, reducing the power usage by up to 62.8% when comparing maximum performance and maximum battery life modes.

Table 2. GPU power usage and clock speed ranges as measured by OCCT and OpenHardwareMonitor.

Mode	**GPU average power usage (W)**	**Clock speed range (MHz)**
Maximum battery life	8.3	350-650
Balanced	21.6	350-1250
Maximum performance	22.3	350-1250

Intel HD Graphics Control Panel is not available under Linux, meaning that there is no easy way to control the power used by the GPU. By default, the Intel HD GPU runs at its maximum speed, being roughly equivalent to the maximum performance mode of the control panel found on Windows. This does mean that the power usage is also increased and battery life decreased. This is non-ideal for situations where maximum GPU performance is not explicitly needed. Such workloads include taking notes in a lecture, reading a document, navigating simple webpages and instant messaging.

2.2.3 GPU induced overheating

Most Intel CPU-s have an on-die graphics solution that provides basic 2D and 3D graphics workload performance. The GPU portion of the CPU is positioned on the same die, as shown on Figure 3.

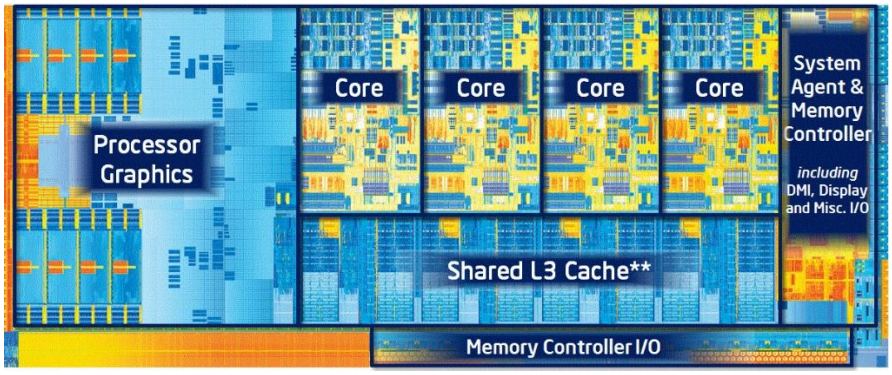

Figure 3. Intel Ivy Bridge microarchitecture die layout. Image released by Intel for marketing purposes, image copy taken from AnandTech [6].

As demonstrated in section 2.2.1 of this thesis, the GPU portion of the package can utilise a considerable amount of power and thus contribute significantly to the total heat output. Due to the positioning of the integrated graphics, the CPU cores closest to the GPU can be affected by the sudden increase in temperatures, possibly causing an unexpected shutdown if the CPU core temperature is past the critical threshold.

GPU induced overheating can be prevented by limiting the GPU performance so that it never exceeds a certain temperature threshold, such as 80°C on a system with a CPU throttling point at 100°C.

2.3 Current power saving methods

While there is no direct alternative to the control panel found on Windows, there do exist some methods to control the integrated graphics performance and improve power savings. This section introduces one GUI (graphical user interface) application and kernel options for controlling the GPU.

2.3.1 Intel Power Control

Intel Power Control is a GPU power management tool that features sliders for GPU clock speeds, a brightness slider and an automatic GPU throttling feature [7]. In addition to that it includes CPU online state toggles and information about CPU package temperatures. The GUI (graphical user interface) component of the application makes use of Python 3 and Qt 5, a framework for building graphical applications, while the helper program is written in C.

While it offers a lot of options and sliders, as visible on Figure 4, it has no recommended set of settings that is found in the Windows program for controlling the GPU performance, making it potentially confusing for less experienced users. In its current state the program requires installing necessary dependencies and building the application yourself, further complicating the install procedure.

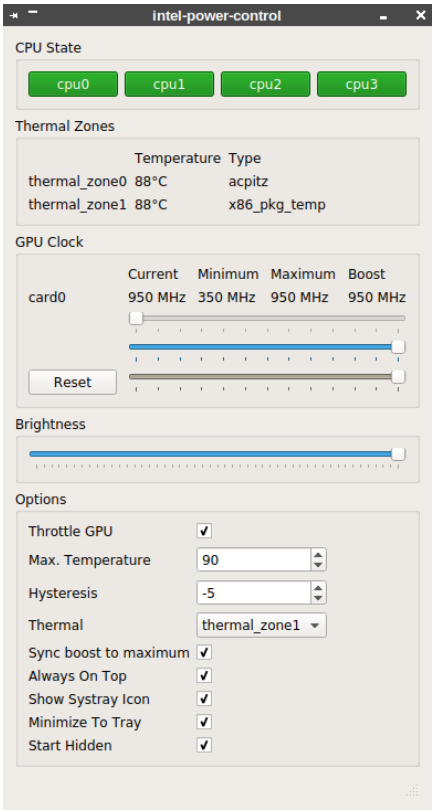

Figure 4. Screenshot of the Intel Power Control application.

2.3.2 Kernel driver options

Intel GPU-s support various methods that can increase power savings and can be configured on the kernel driver level. These include framebuffer compression, LVDS downclocking, enabling RC6 low power states and PCI-e power management [8]. Such features may work on some systems, but can cause problems on other hardware or cause unwanted power usage increases, as demonstrated in a test carried out by Michael Larabel at Phoronix where some options had a negative effect on power usage when running different workloads [9]. The increase in power usage was mostly caused by an increase in performance, as shown on Figure 5.

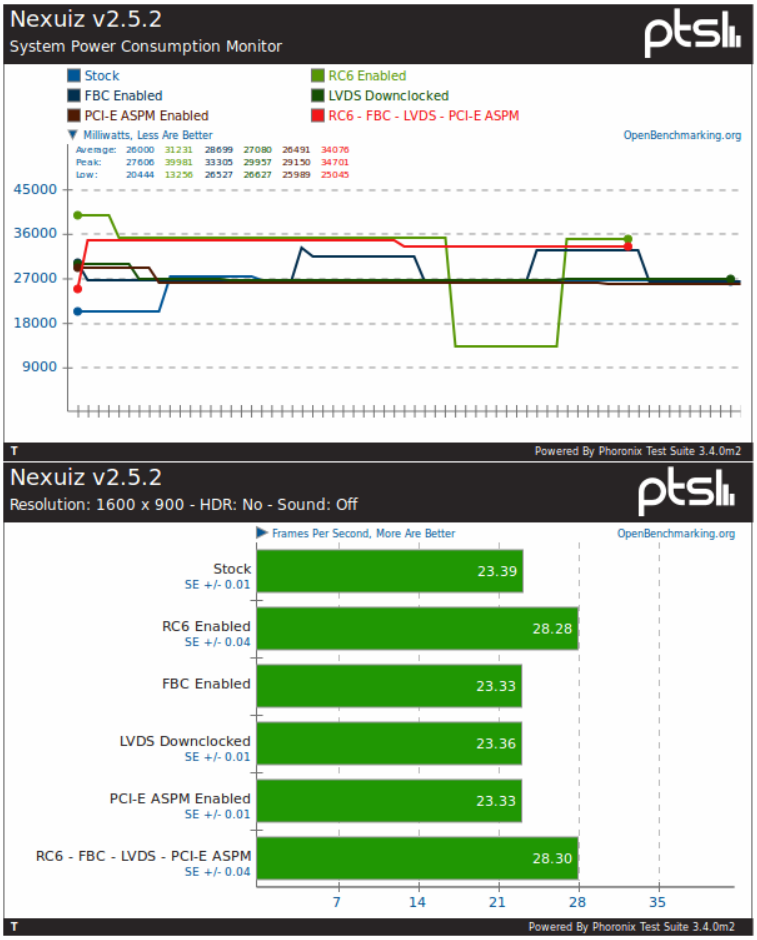

Figure 5. Effect of kernel driver options on performance and power usage with a 3D workload, as tested by Michael Larabel at Phoronix [9].

Since then some features mentioned above have been enabled by default on supported hardware, such as RC6 sleep states. However, even more power can be saved by simply limiting the speed the GPU can run at. This can be achieved by manipulating GPU related `sysfs` paths. For example, on a system with Intel HD 4000 GPU the controls are located at `/sys/class/drm/card0/`. The driver allows to change the minimum, maximum and boost frequencies in steps of 50MHz by writing the proper value to `gt_min_freq_mhz`, `gt_max_freq_mhz` and `gt_boost_freq_mhz`. In addition to that the current active frequency can be read from `gt_cur_freq_mhz` and stock frequency steps from `gt_RP0_freq_mhz` (boost clock speed), `gt_RP1_freq_mhz` (maximum clock speed) and `gt_RPn_freq_mhz` (minimum clock speed), as shown on Figure 6.

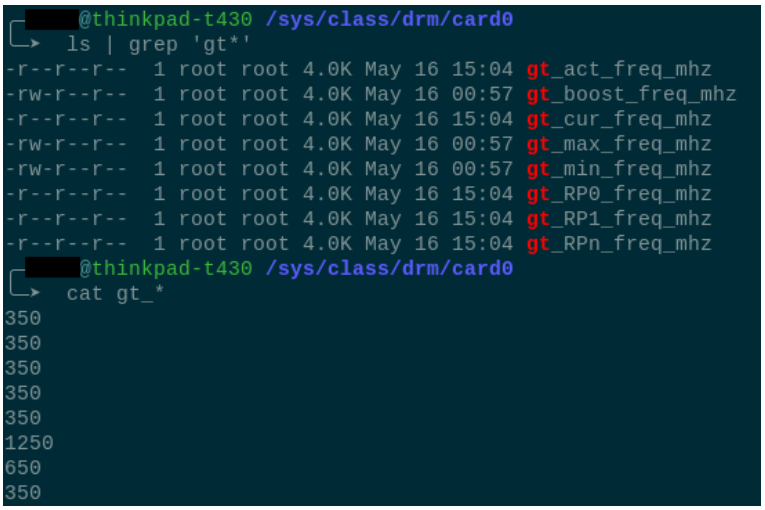

Figure 6. Contents of i915 driver related controls found in sysfs. Commands are run on a PC with Intel HD 4000 graphics running Linux kernel version 4.16.8.

2.4 Linux GPU manager implementation

This section gives an overview of the implementation of the GPU manager solution, covering the architecture, the GPU managing algorithm and an overview of the supported hardware.

2.4.1 Overview

Linux GPU manager is a service that allows any D-Bus capable client application to change the current operating mode. Currently the three modes try to emulate the options provided by Intel HD Graphics Control Panel by offering power modes designed for maximum battery life, balanced (good performance with moderate power usage) and maximum performance.

The current implementation comes with a service with three governor implementations and a simple CLI (command-line interface) program that demonstrates the ability of a client to call a method on the service. When the service is running, the client can set a suitable power mode by giving the power mode name as an argument. An example of the program in action is shown on Figure 7.

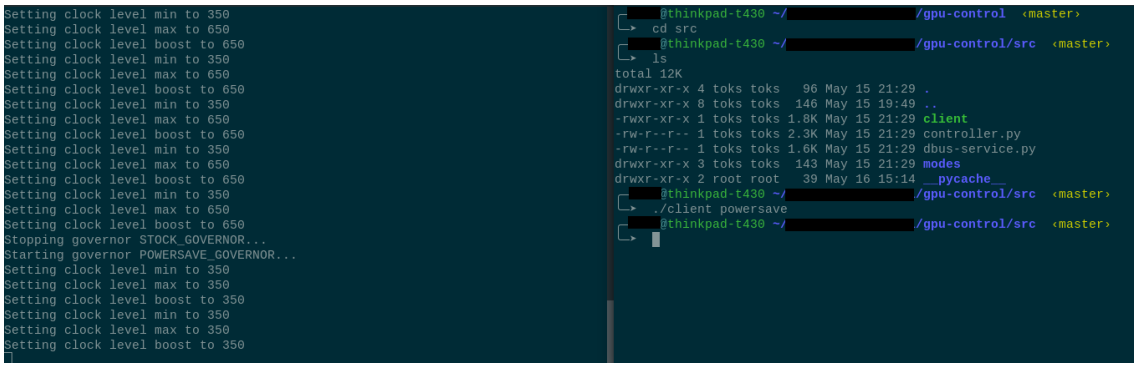

Figure 7. Example of GPU manager client setting the governor from the default one to “powersave”.

Due to the choice of test machines by the author the current implementation of the GPU manager is confirmed to support Sandy Bridge and Ivy Bridge generation integrated graphics solutions. The implementation may already support newer Intel GPU-s, but that hasn’t currently been tested and its effects haven’t been measured. This issue can be alleviated with access to newer hardware and feedback from future users of this solution. An explanation of the i915 driver suggests that the newer GPU models have similar path names with the only change being the RPe frequency which marks the most efficient frequency, being the equivalent to RP1 frequency found on current test hardware [13].

Linux GPU manager is free open source software licensed under GPLv3 [10] and available on GitHub at https://github.com/Hermanio/linux-gpu-manager.

Packaging the program so that it can be used in popular Linux distributions, such as Ubuntu, Debian, Fedora, Arch Linux and others, is going to be implemented in the future along with the GUI component and configuration file support, as these features are not in the scope of this thesis.

2.4.2 Technology overview

This section describes the choice and reasoning behind software components used in the implementation of the GPU solution.

For implementing the server and client portions of the proposed solution, Python 3 was chosen for the programming language. Since all major Linux distributions come with Python 3 already installed, the solution can run out of the box on most Linux installations. Development is also much faster with Python 3 as it makes a lot of tasks very simple with its vast standard library and simple but powerful syntax.

D-Bus is a software bus that allows for IPC (inter-process communication), meaning that multiple concurrent processes can send and retrieve information and signals between each other via the software bus [11]. This is useful for implementing a server-client solution where there exists one server and one or many separate client implementations. This was chosen for the IPC part of the implementation due to it being present in most Linux distributions, and being easy to work with as it has existing language bindings for Python 3.

`sysfs` is a virtual file system that provides access to information about various parts of the kernel, including hardware devices and device drivers [12]. This provides access to the Intel GPU i915 driver. Using the interfaces provided we can control the GPU performance with ease by limiting the allowed clock speed frequency range. Controlling the performance is achieved by writing a value to a virtual file which then passes the value to the proper method in the driver. An example of writing various clock speed values for boost frequency limit is shown on Figure 8.

```
[root@thinkpad-t430 ~]# echo 600 > /sys/class/drm/card0/gt_boost_freq_mhz
[root@thinkpad-t430 ~]# cat /sys/class/drm/card0/gt_boost_freq_mhz
600
[root@thinkpad-t430 ~]# echo 350 > /sys/class/drm/card0/gt_boost_freq_mhz
[root@thinkpad-t430 ~]# cat /sys/class/drm/card0/gt_boost_freq_mhz
350
```

Figure 8. Example of writing values to i915 driver related sysfs files.

2.4.3 Architecture

The architecture of the implementation consists of two parts: a D-Bus service that handles the GPU management, and a client program that can interact with the service using various D-Bus methods. Different behaviours, such as power saving and performance mode, are implementations of a common class called Governor. Diagram of the architecture is shown on Figure 9.

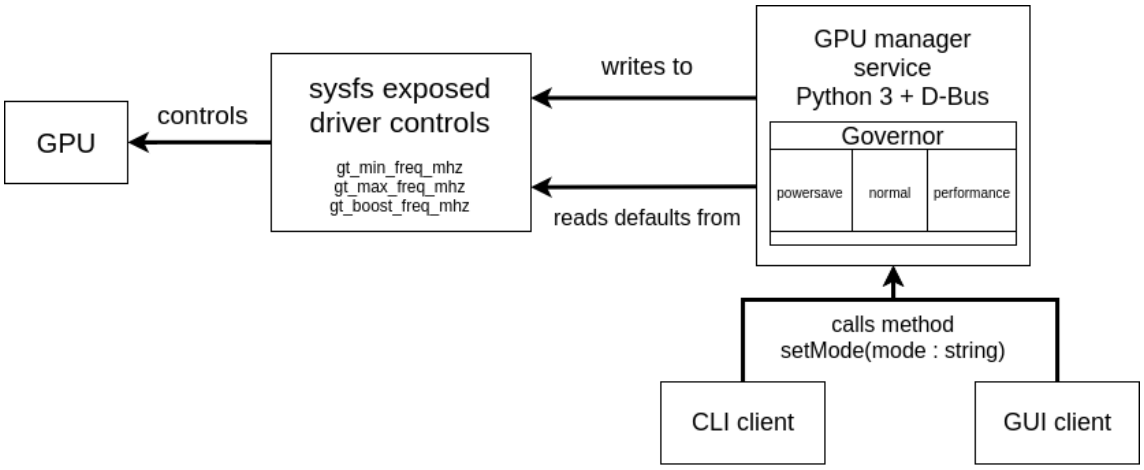

Figure 9. Architectural overview of the GPU manager solution.

Access controls for the client are defined by the D-Bus service configuration file. In its current state the service allows any user to call the `setMode()` method which allows for a non-administrator user to control the GPU power mode. The D-Bus service itself runs on System Bus, giving it root access that is required for writing the GPU clock speed range values.

2.4.4 Algorithm

The GPU manager implementation service starts the D-Bus service and loads the default governor which limits the GPU clock speed to its most efficient frequency, as noted by an Intel GPU driver developer [13]. The existence of an efficient frequency is also present in a utility called intel_gpu_frequency which is found in intel-gpu-tools [14]. Other governors include the power saver mode, which locks the GPU to its lowest clock speed, and performance mode which allows the GPU to utilise the full frequency range. The program flow diagram is shown on Figure 10.

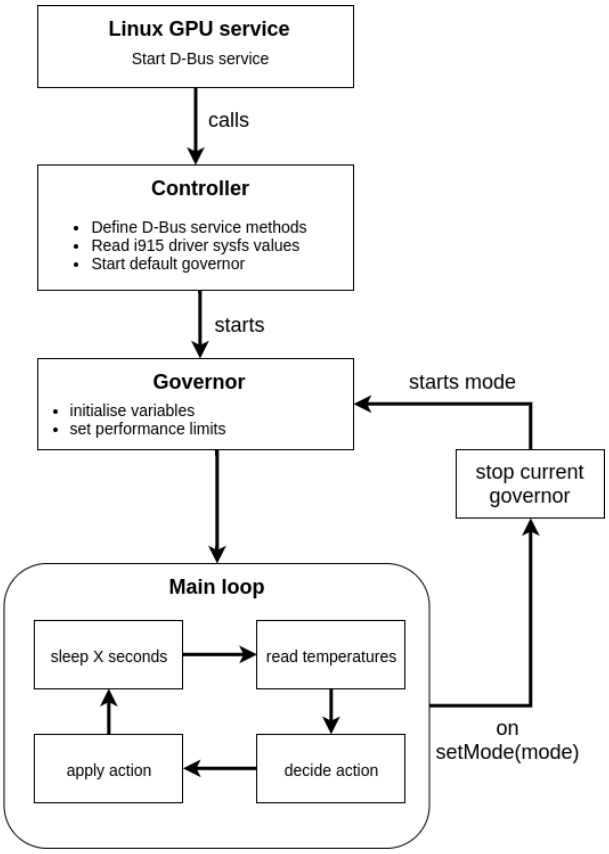

Figure 10. Linux GPU manager program flow diagram.

In addition to simple frequency management the governors can also utilise throttling after a temperature trip point is reached. The temperature is read from the CPU package temperature as it provides the maximum of all temperature sensors on the package. If the temperature is too high, then the GPU clock speed range will automatically be reduced by a multiple of 50 as 50MHz is the smallest allowed change in frequency. The correction depends on the temperature: the higher the temperature the bigger the correction.

2.5 Testing

Testing of the proposed GPU manager solution takes place over 4 tests: heavy 2D workload, heavy 3D workload, 1080p 60FPS video playback and 4K 60FPS video playback. During the tests the power usage of CPU package components (whole package, CPU cores, GPU) and the CPU package temperature is measured. Each test will be run with the GPU solution disabled to get the baseline performance and after that with each GPU manager governor.

Due to the degradation of the laptop battery on the test machine and battery runtime variance that can be caused by other system components during testing, the effect on

battery life is measured using power consumption numbers reported by the CPU package and accessed via `turbostat`, a tool that can measure various CPU metrics, including frequency, power usage, temperatures and C-state residency [15]. This will give a more precise overview of the GPU related energy savings as the `turbostat` program can output package, CPU and GPU power usage separately.

Initially more realistic tests were planned for measuring GPU load in various scenarios using benchmarking tools similar to PassMark BatteryMon and PCMark 8 that emulate real life workloads, including web browsing, office suite work and productivity application interactions. Unfortunately, such tools do not exist yet on Linux and tools that attempt to emulate similar workloads are difficult to compile on some distributions or don't work correctly. One example of such software is gnome-battery-bench which had trouble compiling on Ubuntu 18.04 and relied heavily on an "en-US" keyboard layout and GNOME 3 desktop environment features.

2.5.1 Test device

The testing is done on a ThinkPad T430 with an upgraded quad-core CPU and a liquid metal thermal compound for optimal heat dissipation. The CPU has a TDP of 45W [5] while the T430 is designed for a 35W CPU, such as the i5-3320M [16] , making this test PC a good example of an overheating laptop. Testing is done on a fresh Ubuntu 18.04 install which is booted off an external SSD and connected via an USB 3.0 port. Secure Boot is disabled and UEFI mode is enabled. Specifications are shown on Table 3.

Table 3. Test machine specifications.

Device model	ThinkPad T430
CPU	Intel i7-3820QM
GPU	Intel HD 4000
RAM	16GB DDR3 1333MHz
Display resolution	1600x900
TIM (thermal interface material)	Thermal Grizzly Conductonaut (liquid metal based thermal compound)
Operating system	Ubuntu 18.04 (minimal install)

2.5.2 Overview of tests

Heavy workload testing will be carried out using benchmarks offered by Phoronix Test Suite, an open source testing solution with a vast array of different tests designed to measure all aspects of the PC hardware, including CPU, GPU, networking and storage performance [17]. During the test a temperature and power usage monitoring program `turbostat` will run in the background and log the package temperature and the power usage of the package, CPU and GPU.

2D performance is measured using the test pts/j2dbench which draws various 2D graphical elements, implemented in Java and using the OpenGL API [18].

3D performance is measured using pts/unigine-sanctuary, an older version of the Unigine benchmark software, which puts a heavy load on the GPU by displaying various 3D scenes [19]. Older version of the benchmark was chosen due to the poor 3D performance of the integrated graphics solution when running newer versions of the benchmark.

Video playback testing measures the power consumption during the video playback of a video file while utilising GPU accelerated video decoding. The media files are obtained from http://bbb3d.renderfarming.net/download.html in two formats: x264-encoded 1080p 60FPS (frames per second) and 4K 60FPS. Video playback is done using the `mpv` media player which has hardware acceleration support for video playback enabled on the build found in Ubuntu 18.04, as specified in the Ubuntu wiki [20]. This test can give an idea of power savings during a low to moderate GPU load.

2.6 Results and analysis

This section covers the test results and conclusions that can be drawn from them. In the graphs shown below the different GPU performance modes are marked with a short keyword. Keywords and their meanings are explained on Table 4.

Table 4. Overview of GPU mode names and their descriptions.

GPU mode name	Clock speed range (MHz)	Explanation
stock	350-1250	The default configuration that is applied on a clean Ubuntu install.
powersave	350 (locked)	Limits the performance to the lowest available level as defined by `/sys/class/drm/card0/gt_RPn_freq_mhz`.
normal	350-650	Limits the performance to max efficiency level as defined by `/sys/class/drm/card0/gt_RP1_freq_mhz`.
performance	350-1250	Enables full range of the available GPU performance. Similar to mode "stock".

2.6.1 2D workload test results

Figure 11 shows the effect that the different power modes have on performance. Performance mode matches the stock configuration while modes "normal" and "powersave" have a score that is 45.9% and 68.4% lower, respectively.

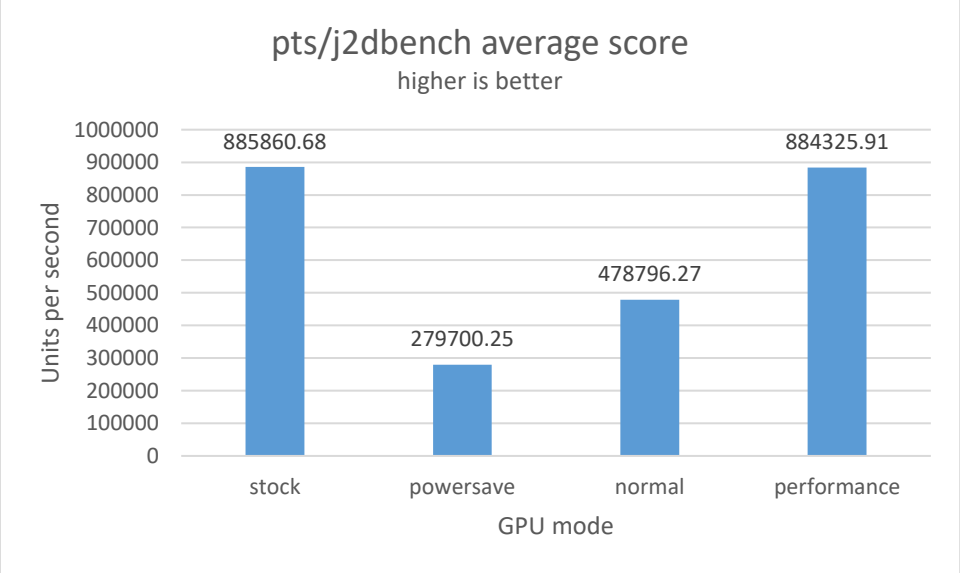

Figure 11. Average scores for the 2D workload test.

Modes "powersave" and "normal" have similar thermal performance, being only 1.57°C apart. Stock configuration and "performance" mode, however, boost the average CPU package temperature significantly, as shown on Figure 12 below.

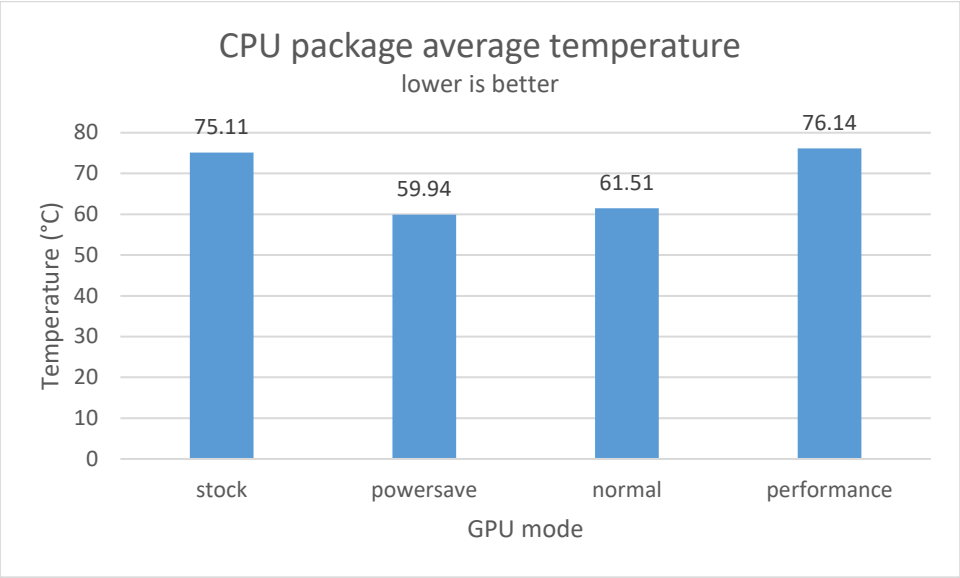

Figure 12. CPU package average temperature during 2D workload testing.

Power usage, which directly affect the CPU package temperatures, shows a similar picture. “powersave” and “normal” modes have the lowest power usage on both the CPU cores and GPU while “stock” and “performance” modes have nearly double or even triple the power usage, as can be seen on Figure 13.

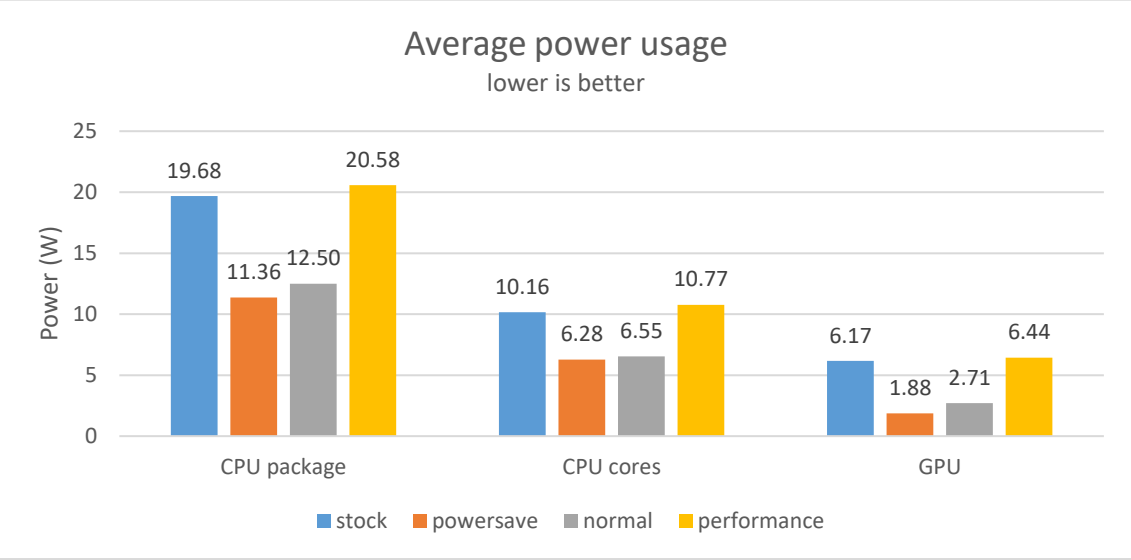

Figure 13. Average power usage of various CPU components during a 2D workload.

2.6.2 2D workload test conclusions

The 2D workload test results clearly show that controlling the GPU power usage can have a huge effect on the power usage not only on the GPU but also on the CPU. This is caused by the increased number of draw calls. The CPU compiles information about the visuals

that are going to be rendered and feeds it to the GPU which then draws the frame [21]. An increased number of frames means more draw calls, meaning more work for the CPU to do and causing the power usage of the CPU to go up. In this particular test the CPU cores power usage can be reduced by 41% simply by going from “performance” mode to “powersave” mode.

When comparing “powersave” and “normal” mode, we can see that the average score is increased by 71% while the GPU power usage increase is only 44.15%. Furthermore, the CPU package temperature is only increased by 2.62% and the package power by 10%, making “normal” mode a good choice for a performance and power usage balanced configuration.

The 2D workload results show that a huge reduction in power usage can be achieved in a heavy workload by simply limiting the GPU performance, making the proposed solution a good choice for scenarios where power savings are more important than performance.

2.6.3 3D workload test results

Performance of the 3D workload is measured in frames per second (FPS). Figure 14 shows that the different GPU power modes have a similar effect on performance when compared to the 2D workload results. Dropping the performance to “normal” and “powersave” modes shows a 39.24% and 66.17% drop in performance, respectively.

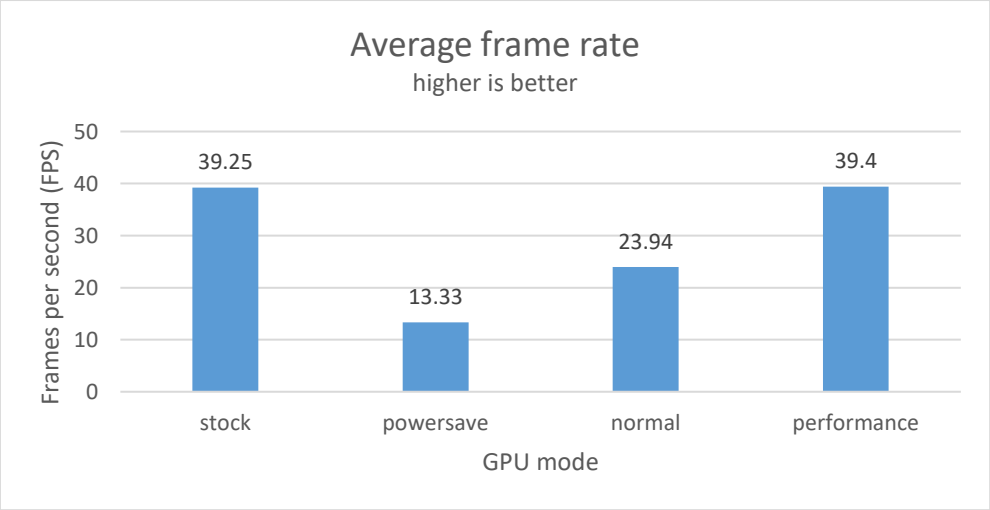

Figure 14. Average frame rate during 3D testing with different modes.

CPU package temperature changes are also similar to the previous test, with the “powersave” and “normal” modes showing the best results with only a difference of

2.57°C. However, the “stock” and “performance” modes show a much higher CPU package temperature, as can be seen on Figure 15, with the “performance” mode nearing 83°C and “stock” mode being at 85.53°C.

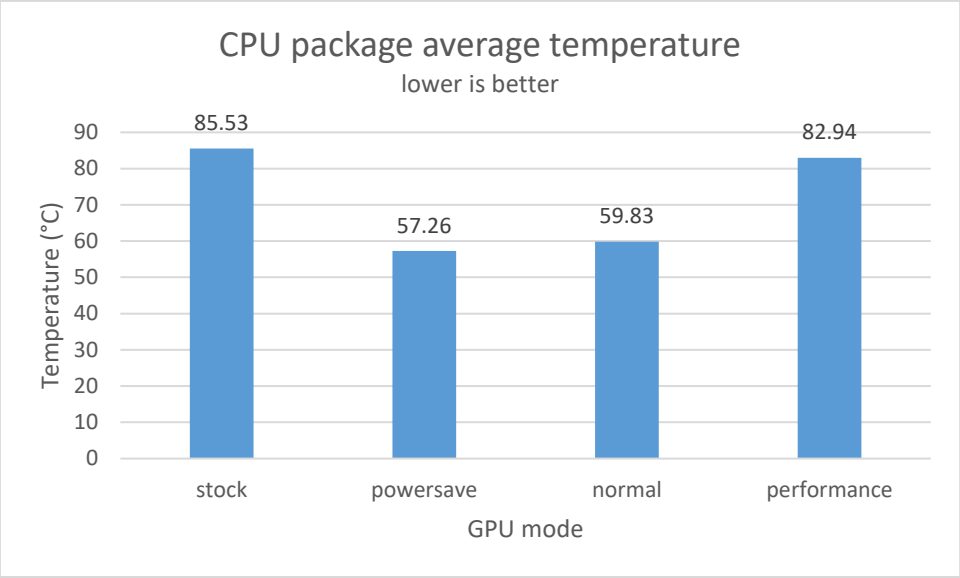

Figure 15. CPU package average temperature during 3D workload.

When comparing the power usage results to 2D workload tests, it is apparent that the power usage numbers have increased, especially for the GPU. For the 3D workload the reduction in performance can bring a power usage decrease of 61.53% or even 77.46% for the GPU. Package power also sees a great reduction as the “normal” mode uses roughly half the power and “powersave” mode a third of the power when compared to “performance” mode, as can be seen on Figure 16.

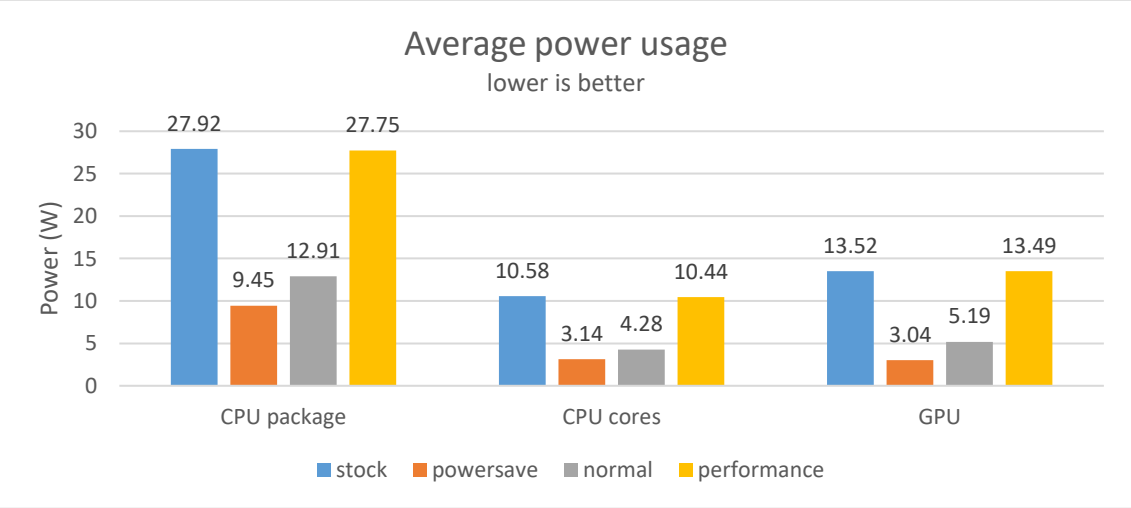

Figure 16. Average power usage for various CPU components during 3D workload testing.

Relative efficiency is calculated by taking the performance indicator (frames per second) and dividing by the average power usage for the CPU package, cores and the GPU, giving us an indicator of the amount of work that the CPU and GPU can do per watt. From the results in Figure 17we can see that both “powersave” and “normal” modes have a greater efficiency for both CPU and GPU components. “normal” mode, which limits the GPU performance to the range of 350-650MHz on the test machine, has the best result which aligns with the statement that the upper limit for this governor is said to be the most efficient frequency for the GPU [13].

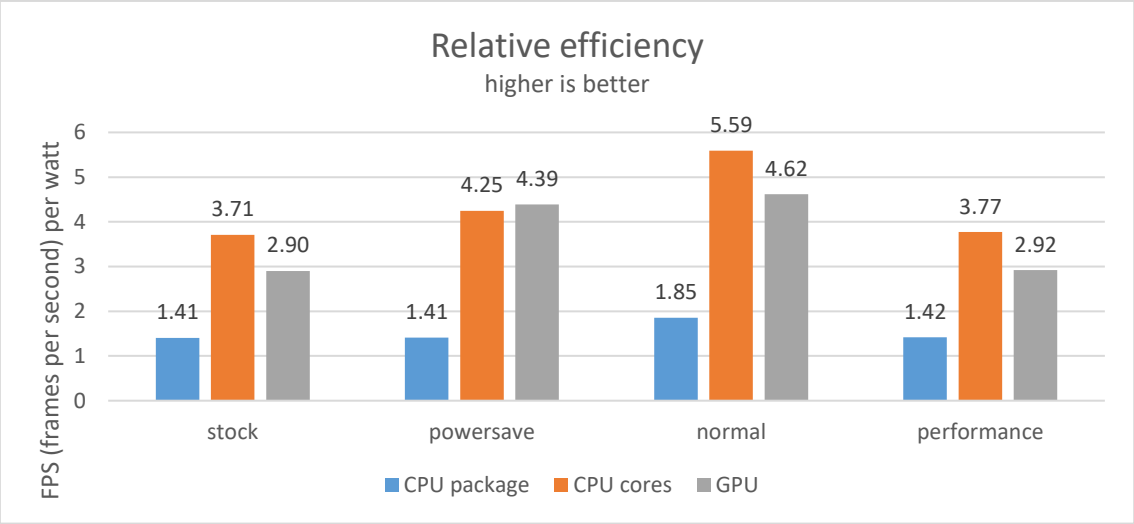

Figure 17. Relative efficiency for the 3D workload over different GPU modes.

2.6.4 3D workload test conclusions

As with the 2D workload testing, limiting the performance in heavy 3D workloads has a noticeable effect for both power usage and CPU package temperature. The effect on efficiency is also noticeable, with “normal” mode having 31.21% better efficiency when compared by the package power and 59.31% when compared by the GPU power usage.

“powersave” mode also sees a boost in efficiency, but the package power efficiency results are less pronounced. This is caused by the various surrounding components of the CPU package that also consume power. When taking the total package power and subtracting the CPU cores and GPU power usage, we get the remainder which is around 3-4W, as shown on Table 5. A smaller relative reduction in power usage does not boost the efficiency indicator as much for the whole package.

Table 5. Breakdown of CPU components power usage during a heavy 3D workload for the stock GPU configuration.

GPU mode	stock
Package power usage (W)	27.92
CPU Cores power usage (W)	10.58
GPU power usage (W)	13.52
Remainder (W)	3.82

As with 2D workload testing, sacrificing performance can greatly increase power savings in situations where it is truly needed. Temperature decrease of 25-27°C also has a great positive impact on user experience as the CPU fan can run at lower speeds and the laptop surface is cooler, making the device more comfortable to use.

2.6.5 1080p video playback test results

Unlike the heavy workloads described earlier, the video playback test shows a difference, but a much smaller one. Figure 18 shows that the temperature difference is minimal, most likely affected by other factors.

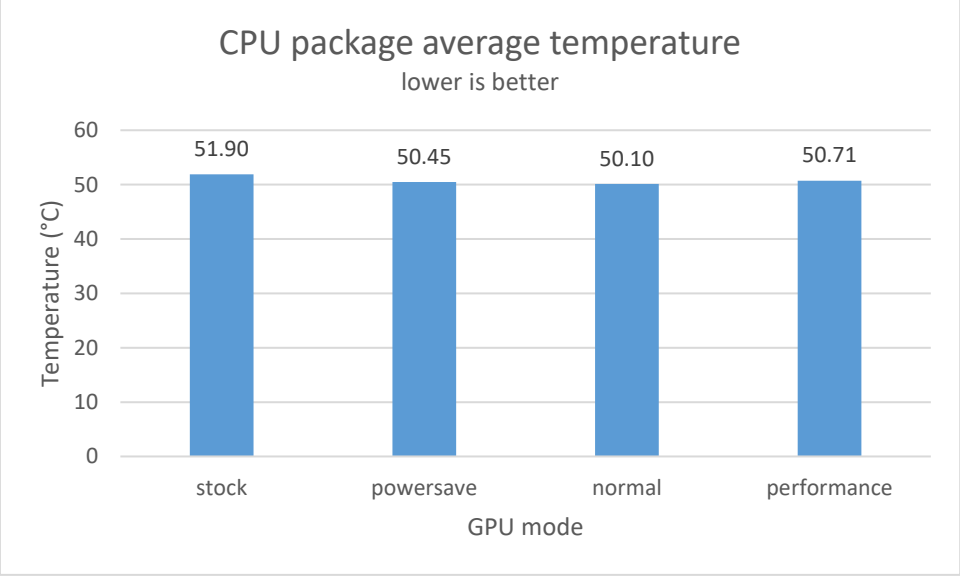

Figure 18. CPU package average temperature during 1080p video playback test.

Power usage also shows small differences, as seen on Figure 19, with “powersave” mode at best reducing the GPU power usage by 0.03W and CPU package power usage by 0.13W. Percentage-wise this is a 2.1% reduction in CPU package power usage.

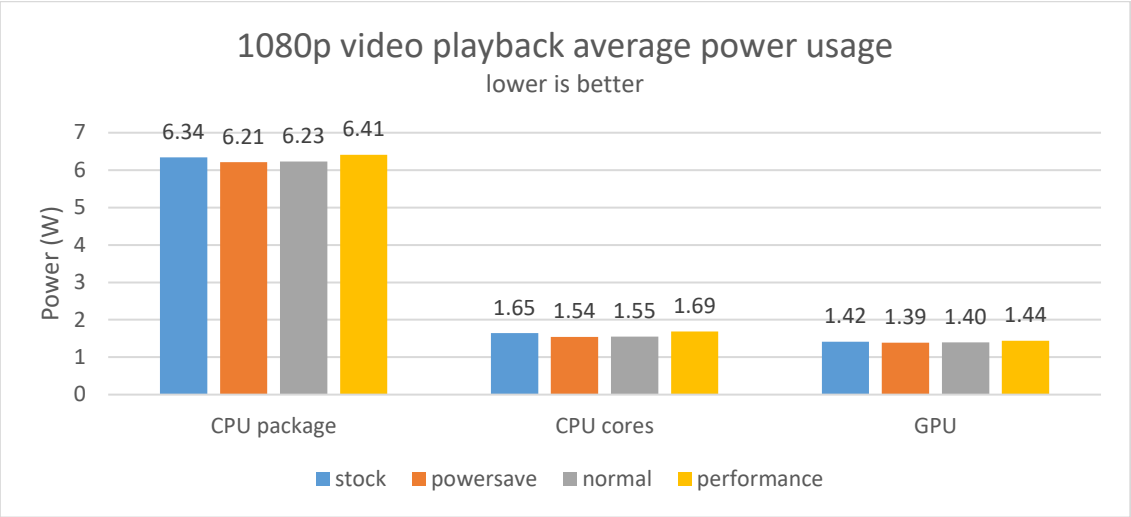

Figure 19. Average power usage of CPU components during 1080p video playback test.

This is most likely caused by the low processing power demands of the video file in question and the stable frame rate that the video offers, allowing the GPU to go into the RC6 sleep state more often and limiting the active time of the GPU. Less active time in turn limits the time that Linux GPU manager can apply its effect.

2.6.6 4K video playback test results

4K 60FPS video playback shows a more pronounced difference in CPU package temperatures, with the “powersave” and “normal” modes giving the best results by having an average CPU package temperature that is 3°C lower when compared to the stock configuration, as evident in the results shown on Figure 20.

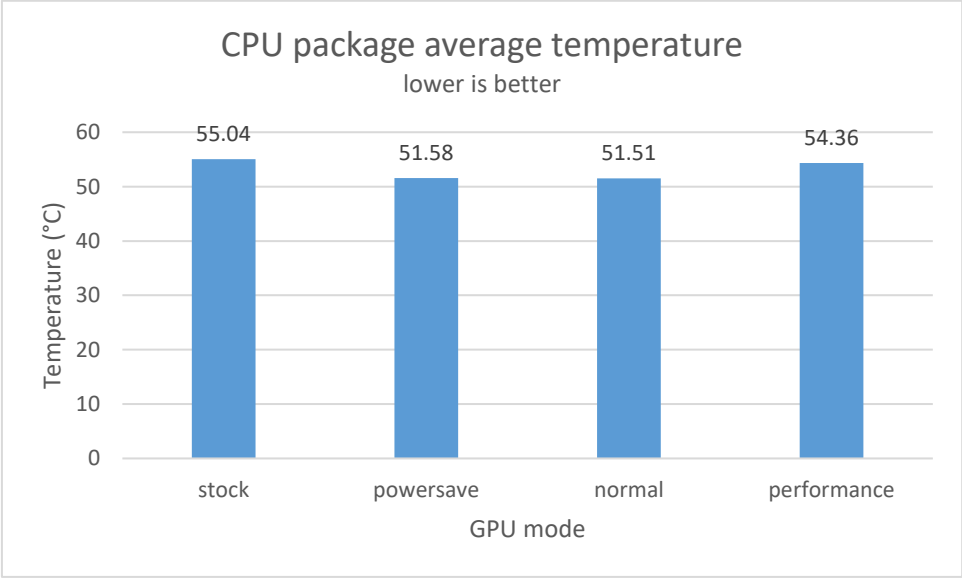

Figure 20. Average CPU package temperature during 4K video playback test.

Comparing the modes by power usage we can see that “powersave” and “normal” mode offer good results, decreasing the CPU package power usage by 18.4%. CPU cores and GPU power usage is reduced by 35.6% and 18.4%, respectively. Figure 21 gives a more visual overview of the power usage numbers.

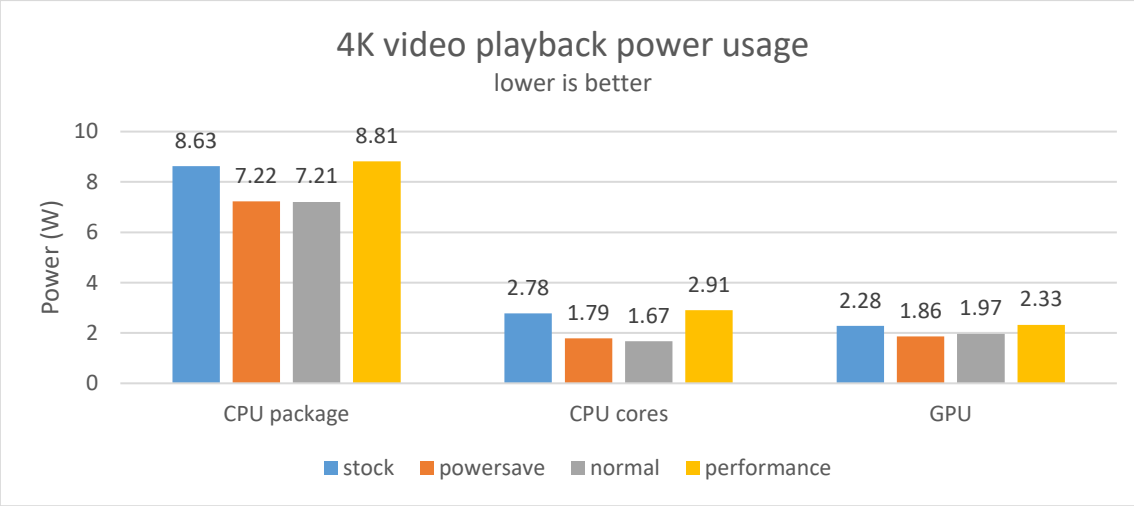

Figure 21. Average power usage of CPU components during 4K video playback test.

While the “stock”, “normal” and “performance” modes showed no degradation in video playback quality, the “powersave” mode experienced 240 dropped frames during playback. With the video runtime of 636 seconds this amounts to 0.6% of frames being dropped, affecting the user experience negatively.

2.6.7 Video playback test conclusions

As can be seen from the results, on 1080p video playback the power savings are not that significant, mostly due to the low GPU load that the playback caused, allowing the GPU to utilise RC6 sleep states. This means that the GPU was active a lot less and the effect of the GPU clock speed limit is less visible.

The increased load on the GPU and CPU by the 4K video playback the effect of the GPU manager is more visible, as the GPU spends less time in sleep states. However, even in this case the decrease in CPU power usage contributes about twice as much to the power usage decrease than the GPU.

Dropped frames with the “powersave” mode indicate that “normal” mode is the best choice for a balanced experience with good performance and power savings. Small but measurable improvements in power usage indicate that the GPU solution can help save power even in lighter workloads.

3 Linux CPU manager

This section is structured similarly to section 2 of this thesis, containing background information about CPU related issues, technologies, existing solutions, overview of the implementation and finally testing and results.

3.1 CPU related issues

Most CPU related issues that crop up are related to excessive CPU fan noise, overheating and a short battery life. This section goes over these issues in more detail.

3.1.1 Overheating

Most, if not all modern CPU-s have thermal protections implemented on the hardware level. If the temperature is over the set limit, the CPU will begin throttling itself, which reduces performance and power usage to bring the chip temperatures down. On modern Intel CPU-s such throttling can start very late. For example, an Intel Ivy Bridge series CPU will start throttling at 100°C and will instantly shut down at 105°C to prevent further damage to the CPU. On-chip throttling is a great feature, but the throttling point can't be configured by the user and the small thermal headroom carries a high risk of an unexpected system shutoff, which can cause loss of work and file system corruption.

In some configurations the CPU on a laptop can be changed for another model provided that the socket is the same and the CPU is supported by the motherboard. It is common among hardware enthusiasts to replace their dual core CPU-s with quad core models in selected laptops which brings a huge boost in performance, especially in multithreaded workloads due to the doubling of the core count. While it is recommended to pick a CPU with the same TDP as the old one, some users opt for higher TDP models due to pricing or availability of replacements on the used computer equipment market. If the cooling system is designed for a 35W TDP CPU, but the new CPU has a TDP of 45W, heat dissipation will become an issue. On extended workloads the CPU will start throttling and can potentially overheat if the cooling is not sufficient.

3.1.2 Noise and battery life issues

With the increase in CPU temperatures the system must be able to dissipate the heat properly. In most cases this means increasing the fan speed, which increases airflow and effective heat dissipation. Some laptop manufacturers, such as Lenovo and Dell, allow the CPU fan speed to be controlled by the operating system if the necessary drivers are loaded and a special utility is running. Such examples include thinkfan and i8kutils for Lenovo ThinkPad laptops and selected Dell systems, respectively.

However, on most systems there is no straightforward way to directly control the fan speed, meaning that the only way to lower the fan speed is by lowering the CPU temperatures. This can only be achieved by reducing the power usage of the CPU. This approach is perfectly acceptable for workloads that take considerable time to complete, such as video rendering, software compiling and scientific calculations. If these are run during the night, when fan noise is most audible and noticeable, then it will make sense to run the workload slower if it reduces the noise output of the computer and finishes in time.

In less CPU heavy tasks, like writing down notes, reading an article or using a messaging client, very little processing power is needed to complete the task. However, the usual OS installation has a lot of processes running in the background, which includes system services, file synchronization programs, music players and many browser tabs with different levels of CPU usage. The background programs can have a noticeable negative effect on battery life, as they ask for a lot of compute power without any regard for power usage and efficiency. This negative effect could be reduced by limiting the speed the CPU can run at which in turn limits the amount of power the background tasks can use.

3.2 Technical background

This section explains some technical concepts that the implementation and its behaviour is based on. The first half focuses on CPU power usage characteristics and the second half gives an overview of optimal throttling methods.

3.2.1 CPU power usage characteristics

To reduce the CPU temperatures without any additional cooling devices present we need to consider the power usage of the CPU. CPU power usage is linked to the operating

frequency. As the frequencies get higher, the voltage of the CPU must be increased. This increase in voltage is directly responsible for the increase in power usage. The relationship between the operating frequency and voltage is not linear. At lower frequencies an increase in frequency may only mean a small bump in voltage. However, at higher frequencies a relatively modest increase in frequency may need a huge increase in voltage [22, p. 36].

The relationship between voltage and frequency depends on the CPU model, generation and architecture, but generally follows the rule that voltage scales with the square of the frequency [22, p. 36]. Figure 22 shows that in order to increase the frequency at higher frequencies, a much higher increase in voltage is required. An exponential curve means that the most efficient frequency of the CPU is not at the lowest or highest frequencies, but somewhere in-between. The exact point depends heavily on the CPU itself, but can be determined by running tests in each performance state.

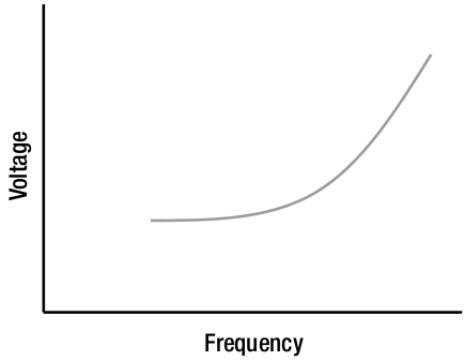

Figure 22. The relationship between voltage and frequency [22, p. 37].

Modern CPU-s have a number of P-states (performance states) which determine the CPU frequency and voltage. P-states are usually managed by the operating system and are in steps of base clock frequency which is commonly 100MHz on modern Intel and AMD CPU-s [22, pp. 49-50]. Limiting the number of available P-states can reduce CPU temperatures and power usage because voltage is also reduced. Furthermore, the efficiency of the CPU can also be improved because the operating system can't request higher performance states which are less efficient.

3.2.2 Optimal throttling method

The default throttling behaviour of a CPU relies on a trip point at which it starts throttling, commonly 100°C but can vary between models. At 100°C the CPU will attempt to reduce

temperatures by limiting power usage and performance. Figure 23 shows an example of the effect of CPU thermal throttling on frequency. During the load the CPU temperatures are generally in the range of 100-102°C.

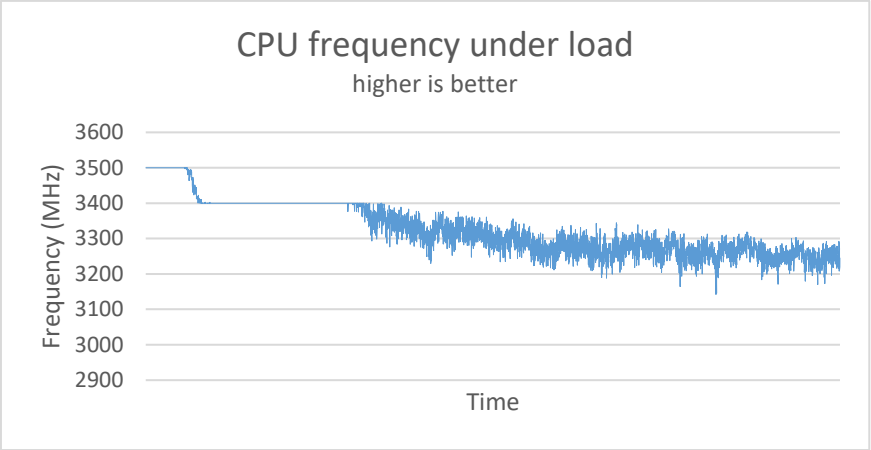

Figure 23. Effect of CPU throttling on CPU frequency on a test system with Intel i7-3820QM CPU.

Previous work on CPU cooling and optimal throttling methods has found that running the CPU near a set temperature limit is the most optimal throttling method as it provides the best performance compared to alternative throttling methods. This is backed by [23] where system-throttling, a method of throttling where the CPU clock speed is reduced when over the temperature limit and increased when under it, provided better performance to alternative throttling methods. Test results in [24] also demonstrate that slightly increasing or decreasing the clock speed near the temperature limit yields better performance compared to an alternative throttling method while keeping the temperatures in control. A similar method of throttling has also been successfully used on ARM-based CPU-s in previous work where the authors made use of existing Linux infrastructure to throttle the CPU when it reached a set temperature limit, such as 50°C [25].

3.3 Intel CPU throttling methods on Linux

Several effective throttling methods exist on Linux based operating systems. This section introduces the Intel P-state driver, Intel Powerclamp and RAPL (Running Average Power Limit) controller that can all be used to limit the CPU performance and power usage.

3.3.1 Intel P-state Driver

Intel P-state driver is a CPU hardware driver and governor that manages the power states of the CPU. It is enabled by default on white-listed CPU-s from the Sandy Bridge generation onwards [26].

P-state driver offers two governors: “powersave” and “performance”. These are accessible through CPUFreq subsystem and chosen by writing the value to `/sys/devices/system/cpu/cpu0/cpufreq/scaling_governor`. Performance governor makes the CPU always choose the highest P-state with little care for energy usage. “powersave”, however, attempts to balance performance with energy savings [26].

Voting for a P-state among cores which share the same voltage domain is simple: highest P-state among a set of cores wins. For example, if on a dual core system one core requests the lowest P-state and the other core a higher P-state then the higher P-state will be applied for both cores [22, p. 51].

Minimum and maximum P-states can be controlled via `sysfs` interface on path `/sys/devices/system/cpu/intel_pstate` by writing the value to `max_perf_pct` and `min_perf_pct` [22, p. 288].

In this example case for a ThinkPad X230 with an Intel i5-3320M CPU the `intel_pstate` driver exposes various information about maximum and minimum performance states, turbo mode status and information about the number of performance states and the turbo mode start percentage, as shown on Figure 24.

```
root@dev-testbench:/sys/devices/system/cpu/intel_pstate# cat *
100
36
0
22
active
33
root@dev-testbench:/sys/devices/system/cpu/intel_pstate# ls -l
total 0
-rw-r--r-- 1 root root 4096 apr   26 22:17 max_perf_pct
-rw-r--r-- 1 root root 4096 apr   26 22:17 min_perf_pct
-rw-r--r-- 1 root root 4096 apr   26 22:17 no_turbo
-r--r--r-- 1 root root 4096 apr   26 22:17 num_pstates
-rw-r--r-- 1 root root 4096 apr   26 22:17 status
-r--r--r-- 1 root root 4096 apr   26 22:17 turbo_pct
root@dev-testbench:/sys/devices/system/cpu/intel_pstate#
```

Figure 24. Intel P-state driver controls and their values.

If we set the `max_perf_pct` value to 36 (default value of `min_perf_pct` for this CPU) during a heavy load then we can observe a huge drop in CPU frequency and power usage, as is visible on Figure 25. The software used for demonstrating the effect is `s-tui`, a CLI (command-line interface) application that can display CPU frequency, power usage, utilisation and temperature. In addition to that `s-tui` can also be used to stress test the CPU which is useful for showing the effect of changing performance states.

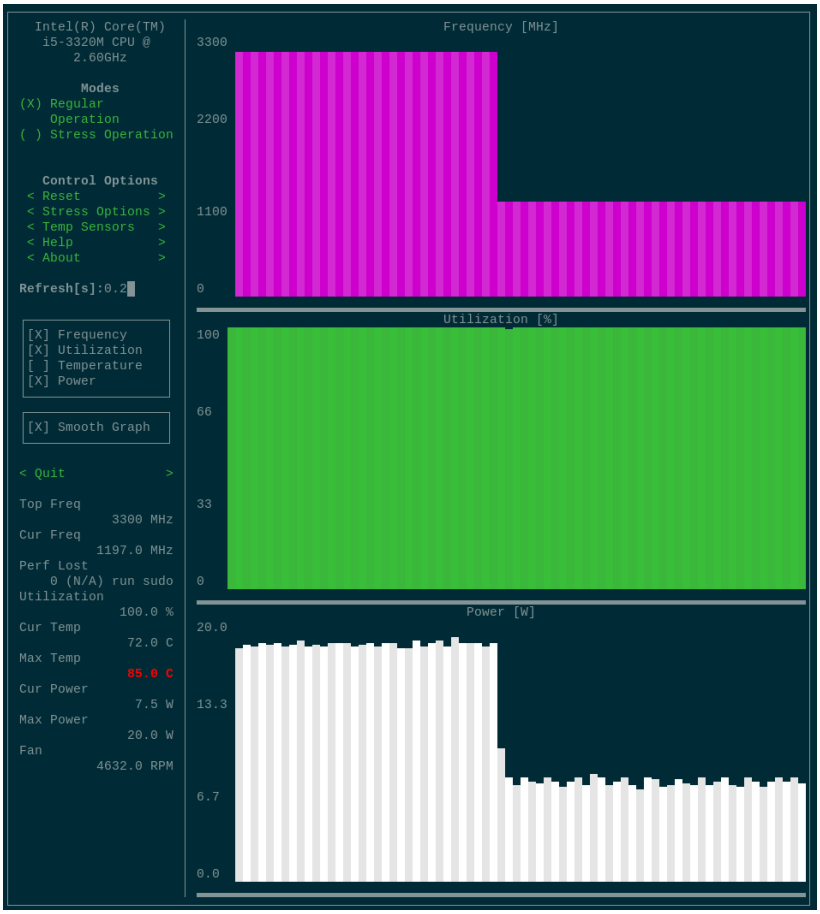

Figure 25. Screen capture of s-tui showing the effect of limiting CPU P-states to its minimum value using intel_pstate.

From Figure 25 we can observe that the clock speed is set to the minimum clock speed for that CPU (1200MHz) and with that the power usage is also dropped to an average of 7W from about 18W previously.

The `intel_pstate` driver also exposes an interface for controlling turbo mode with the `no_turbo` interface. By writing the value "1" to `/sys/devices/system/cpu/intel_pstate/no_turbo` we can turn off the turbo boost functionality of the CPU and run it at its rated base clock speed which locks this particular

CPU to 2.6GHz and lowers the power usage to around 13W in our example case, as seen on Figure 26.

Figure 26. Screen capture of s-tui demonstrating the effect of disabling turbo mode using intel_pstate driver.

With the simple controls offered by the `intel_pstate` driver and the immediate power usage results it is a good candidate for user-defined thermal throttling.

3.3.2 Intel Powerclamp driver

Intel Powerclamp driver is a method of injecting idle time to the system in order to reduce performance and power usage [27]. The driver accepts input as a percentage of idle time, from 0 to a maximum of 50%. The driver can be accessed as a separate cooling device under the path `/sys/class/thermal/cooling_deviceX`, where X is an integer.

In this example the Intel Powerclamp driver is under `/sys/class/thermal/cooling_device5` and can be enabled by writing an integer to `cur_state` with the command "`echo 50 > /sys/class/thermal/cooling_device5/cur_state`".

As we can see from Figure 27, the average power usage dropped by half, indicating that Powerclamp works as intended. There do exist limitations, as mentioned in the kernel documentation, regarding interrupts, which may make the Powerclamp method less effective. Furthermore, the observed CPU power usage is a bit higher when compared to the effect that the Intel P-state driver has.

Figure 27. Screen capture of s-tui demonstrating the effect of Intel Powerclamp with 50% idle time.

3.3.3 RAPL Controller

Running Average Power Limit is a feature on Intel CPU-s from Sandy Bridge generation onwards that offers power monitoring and controlling tools that can be used to measure currently used power, maximum and minimum power supported for domains (package, core devices, DRAM) and methods to set power limits in short term or long term [28].

The general ideal of RAPL is to offer maximum performance provided that CPU thermals and power delivery allow for it. It does this by defining different power levels which have a time constant and a power limit value associated with it [22, pp. 58-59]. For example, a

CPU with a TDP of 45W can use 56.25W for a very short time period (0.000977 seconds for Intel i7-3820QM) and 45W over a longer time period (28 seconds for Intel i7-3820QM).

Power usage monitoring is useful for determining CPU package power usage during various workloads. Setting CPU short term and long-term power limits is helpful in situations where a lower power usage is needed. On Linux this can be achieved with tool `rapl-set` which can set various constraints and power limits [29]. For example, for setting a short-term power limit of 40W over the period of 1 millisecond for constraint 0 the command would be "`rapl-set -c 0 -l 40000000 -s 1000`".

However, this feature is not available on every platform, as some vendors lock the values on the BIOS level. To check for this, load the `intel_rapl` kernel module with "`modprobe intel_rapl`", then try enabling the feature or setting a limit. If `rapl-set` returns an error then the limits are most likely locked, as illustrated on Figure 28.

```
root@dev-testbench:/sys/devices/virtual/powercap/intel-rapl/intel-rapl:0# rapl-set -c 0 -l 10000000
Error setting power limit: No data available
Considerations for common errors:
- Ensure that the intel_rapl kernel module is loaded
- Ensure that you run with administrative (super-user) privileges
root@dev-testbench:/sys/devices/virtual/powercap/intel-rapl/intel-rapl:0# dmesg | grep powercap
[ 3291.117454] powercap intel-rapl:0: package locked by BIOS, monitoring only
[ 5742.746059] powercap intel-rapl:0: package locked by BIOS, monitoring only
[12761.632603] powercap intel-rapl:0: package locked by BIOS, monitoring only
root@dev-testbench:/sys/devices/virtual/powercap/intel-rapl/intel-rapl:0#
```

Figure 28. Demonstration of an attempt to change power limit values and timings on a system with those values locked.

Some laptops, such as ThinkPad T430 and ThinkPad X230, have the limits locked and unavailable for modification. As such using Intel RAPL for controlling and limiting power usage for throttling is not guaranteed to work on all machines using Intel CPU-s, making it a non-ideal candidate for a generic throttling method. The monitoring methods are still available and can be useful for measuring the power usage of the CPU package, including the CPU cores and GPU separately.

3.4 Existing solutions

To overcome the CPU related issues mentioned in this thesis, some solutions have been already created. Performance controlling solutions, such as scripts and extensions, and pre-emptive thermal throttling software are covered in this section.

3.4.1 Simple scripts and GUI applications

Current solutions for controlling the CPU behaviour are generally simple scripts of varying quality that can be found on GitHub or GUI applications in the form of desktop environment extensions and GTK+ or Qt based graphical applications. The solutions are simple in functionality as they simply write the values once and offer no other features or recommendations for a less experienced user. Furthermore, the presentation layer and the logic that does the actual CPU controlling are heavily tied together in the case of extensions and GUI applications.

Some solutions are also very limited in their support. For example, running a GNOME 3 extension is only possible on that particular desktop environment, effectively leaving out users of other desktop environments, such as KDE, MATE, XFCE and others.

One example of a GUI solution is a GNOME 3 extension made by Martin Koppehel called "CPU power manager" that allows for configuring the minimum and maximum CPU power states [30]. The interface is shown on Figure 29.

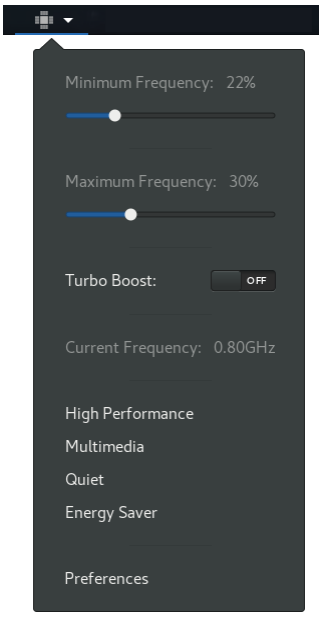

Figure 29. Screen capture of a GNOME 3 CPU power manager extension [31].

Using this extension, it is possible to configure various profiles and set different performance levels. It does help with managing power usage and temperatures, but unfortunately this extension is only available on GNOME 3 and it provides no throttling functionality.

3.4.2 Linux Thermal Daemon

Linux Thermal Daemon is an open source project by Intel that aims to offer a user mode daemon for thermal management. Its description mentions that it is aimed towards system developers who manage thermal dissipation in various configurations, including desktops, laptops, smartphones and embedded devices [32].

The project makes use of existing kernel infrastructure and aims to keep the temperatures under control while maintaining optimal performance. It does so by getting input data from various sensors and activating various cooling devices after a trip point has been reached [32].

Cooling devices include performance states (P-states), the RAPL (Running Average Power Limit) controller, Powerclamp driver, T-states and additional cooling devices, such as fan controllers, that can be managed from the configuration file.

With the default configuration the thermal daemon reads temperature information from `/sys/devices/platform/coretemp.0` and calculates a temperature trip point using `temp1_max` and `temp1_crit` values. For example, Intel i5-3320M reports `temp1_max` as 87000 and `temp1_crit` as 105000 (87°C and 105°C), resulting in a trip point of 96°C.

As observed by the author in some hardware configurations and demonstrated in Figure 30, the behaviour of Linux Thermal Daemon can be erratic and have negative consequences, such as hardware damage due to overheating and jumps in performance manifested by stutter, dropped frames, low responsiveness.

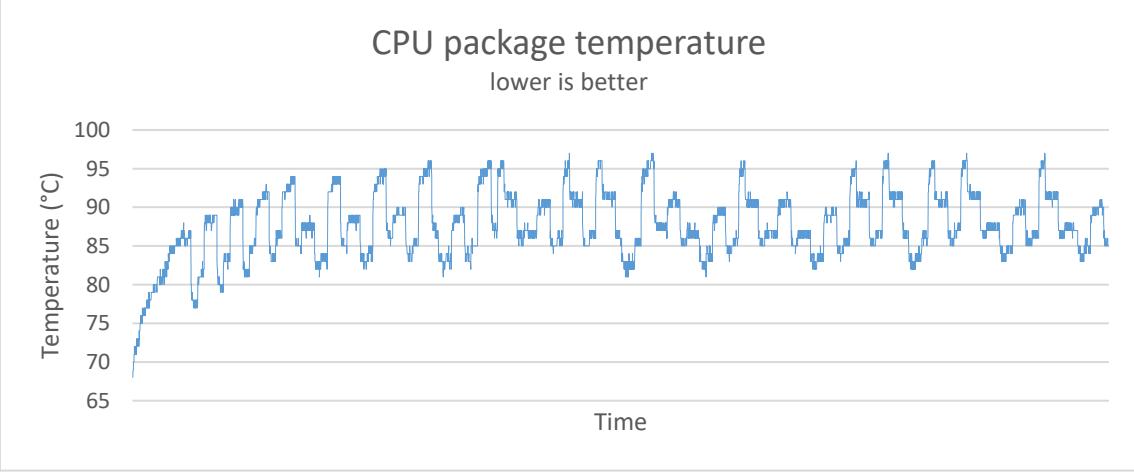

Figure 30. Example of erratic Linux Thermal Daemon behaviour on the test machine. The target temperature was set to 87°C in the configuration file.

3.5 Linux CPU manager implementation

This section goes over the implementation of the CPU manager solution, going over the technology used, the general architecture, different governors offered and details about the throttling algorithm and hardware support.

3.5.1 Overview

Linux CPU manager is a D-Bus service that allows the user to limit the available CPU performance to achieve better thermals, increase efficiency and improve battery life. The service offers multiple CPU governors that limit the performance and throttle the CPU automatically to prevent overheating. The client application can control the service using the specified API.

The general structure of the CPU manager is based on Linux GPU manager implementation discussed in section 2.4 of this thesis, meaning that it is written in Python 3 and makes use of D-Bus language bindings and the `sysfs` virtual filesystem. The difference is in the driver it controls and the method of throttling used. The CPU performance is managed using the `intel_pstate` driver by controlling the allowed performance states range and the turbo mode status.

Linux CPU manager is supported on all Linux based operating systems which have the `intel_pstate` driver enabled. In future iterations support for other cooling methods, such as RAPL, Powerclamp and CPUFreq, can easily be added.

Linux CPU manager is a free open source project licensed under GPLv3 [10] and available at https://github.com/Hermanio/linux-cpu-manager.

3.5.2 Architecture

Architecturally the Linux CPU manager is very similar to the GPU manager due to the similar method of limiting performance. CPU manager, however, provides four different governors and controls `intel_pstate` driver controls which can limit the performance states of the CPU and toggle the turbo mode state. Figure 31 shows a diagram of the architecture of the CPU manager.

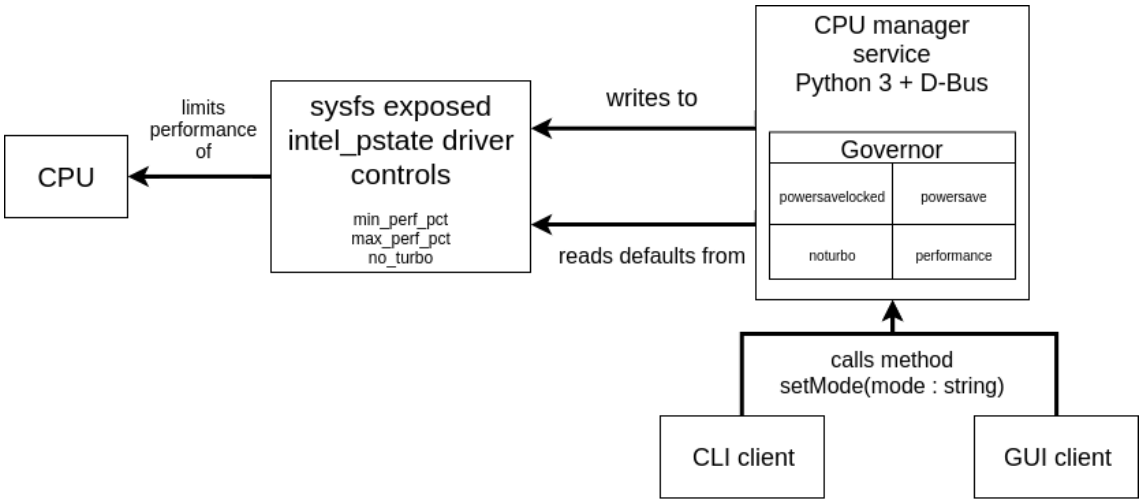

Figure 31. Architecture of the Linux CPU manager.

Linux CPU manager has many different governors that aim to offer choices for running the CPU with minimal power usage, at efficient speeds or at maximum performance with optimal throttling enabled. Their codenames and descriptions are shown on Table 6.

Table 6. Linux CPU manager governors and their clock speed range, scaling governor and description.

Mode	**CPU clock speed range on test CPU (MHz)**	**Scaling governor for intel_pstate driver**	**Description**
powersave locked	1200	powersave	Locks the CPU to the lowest performance level available.
powersave	1200-2000	powersave	Sets the available range from minimum to middle point between lowest and turbo performance level (around 55pct).
noturbo	1200-2700	powersave	Sets the available range from minimum to maximum non-turbo clock speed (also known as the base clock speed). Throttling enabled at CPU max temperature (87°C for this test CPU).
performance	1200-3700	performance	Allows the CPU to run at its full clock speed range.
75°C target, 96°C target	1200-3700	performance	Quick modifications to performance governor to test CPU throttling behaviour at 96°C and 75°C targets. Temporary governors until temperature limit setting and configuration file support is finished.

3.5.3 Algorithm

Linux CPU manager service starts the default governor “stock” which limits the performance of the CPU to its base clock speed, meaning that turbo mode is disabled. A CLI client can be used to change the governor in a way similar to Linux GPU manager.

Each governor has a set polling period: 5 seconds for “powersave locked” and 0.25 seconds for other governors. The loop action flow is shown on Figure 32.

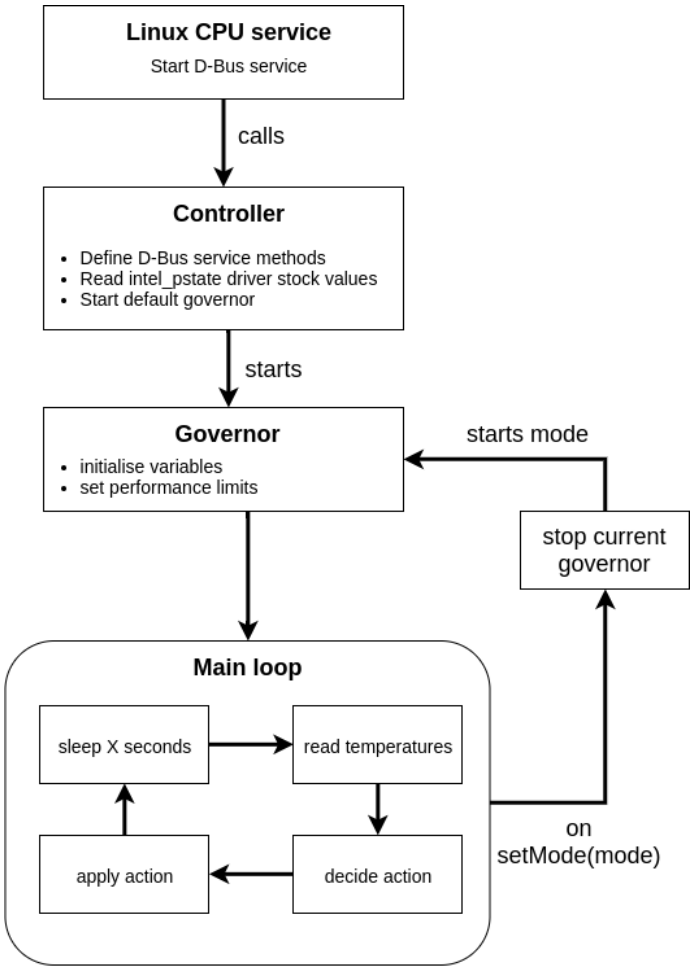

Figure 32. Linux CPU manager service program flow.

3.6 Testing

All CPU tests are done on the test machine described in section 2.5.1 of this thesis. The testing of the Linux CPU manager component focuses on two aspects: the throttling component and the effect on performance, power usage and efficiency of its different governors.

The throttling component is observed while running a CPU stress test with 16 active threads over 10 minutes in different configurations. This gives an overview of the thermal throttling behaviour at different temperature targets and mechanisms. The throttling behaviour is also compared to the stock configuration and Linux Thermal Daemon.

The performance component is measured using the CPU heavy Linux kernel build test provided by Phoronix Test Suite. This gives a comparison of Linux CPU manager and Linux Thermal Daemon performance and the effect of pre-emptive throttling on performance when compared to no pre-emptive throttling. Kernel build test also measures the performance, power usage and efficiency of the different CPU governors.

In the following test descriptions and results Linux CPU manager will be referred to as LCM and Linux Thermal Daemon as LTD.

3.6.1 Thermal throttling tests

When running the CPU without any user-space throttling software running the CPU eventually hits the 100°C temperature limit and starts thermal throttling. The temperatures are stable due to the nature of CPU thermal throttling. However, at some points the temperature can peak at 103°C, being dangerously close to the 105°C critical temperature shutoff point. If a process suddenly put load on the on-die GPU then an overheating induced shutdown would be very likely.

LTD thermal behaviour seems to be very close to LCM when comparing the average temperature. Looking at the temperature graphs shows a different picture, however, as it is very clear that LTD has a dramatic variance in temperatures, as shown on Figure 33.

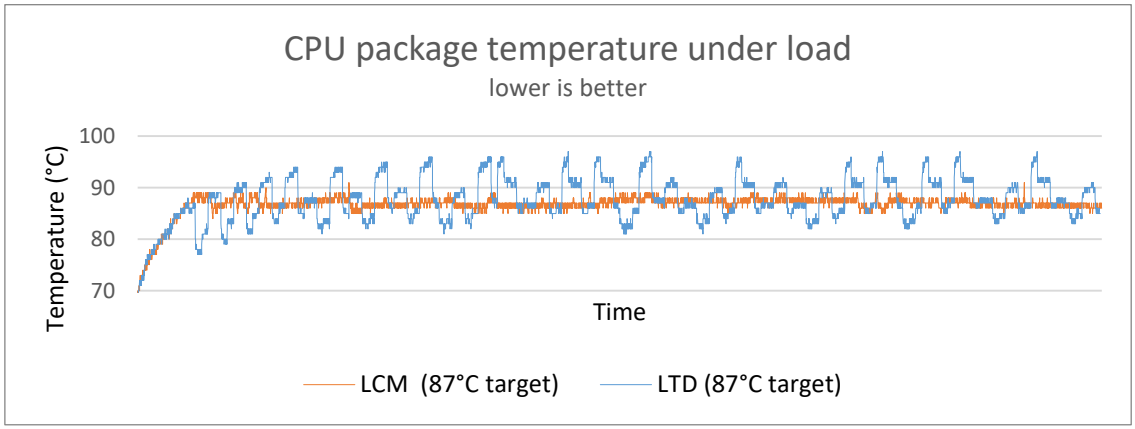

Figure 33. CPU package temperature comparison between Linux Thermal Daemon and Linux CPU manager.

LCM crosses the set temperature limit by 1-3°C, rarely by 3-5°C, while LTD does this very often, going over the limit by 5-10°C, effectively not respecting the temperature limit. In its stock temperature configuration this means going over the 96°C trip point and reaching temperatures of up to 102°C, as visible on Figure 34.

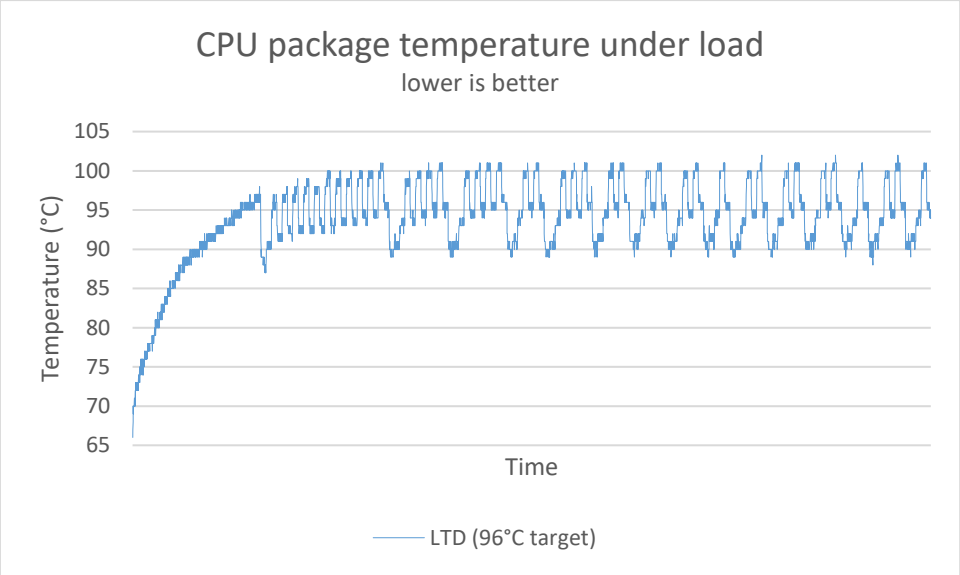

Figure 34. Thermal behaviour of Linux Thermal Daemon with its default settings under load. Note the variance in temperature and failure to prevent hitting temperatures over 100°C.

LCM also has a higher average CPU clock speed, as shown on Figure 35, meaning more work being done with same thermal limit. This aligns with previous research that suggests that running the CPU near its thermal limits is the best approach performance wise.

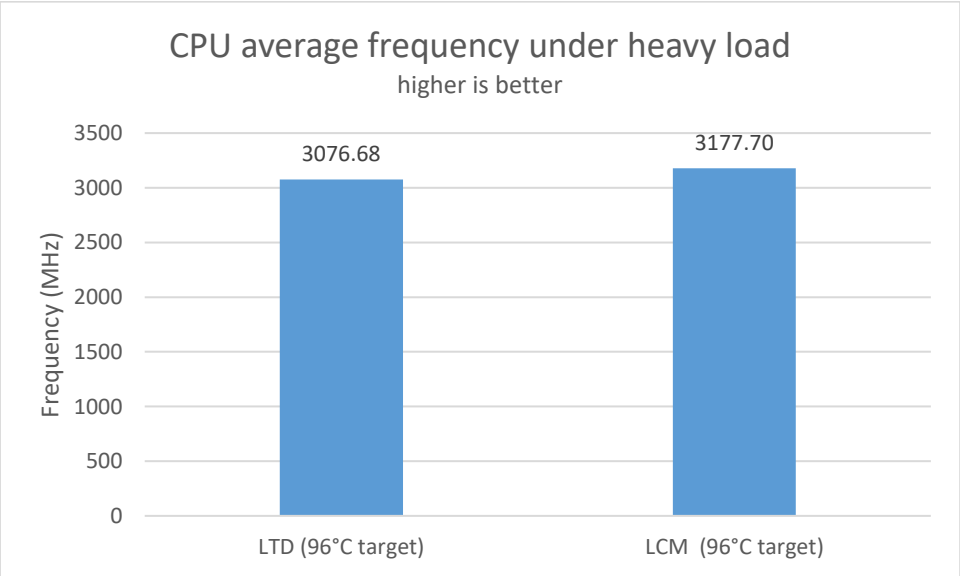

Figure 35. Comparison of average CPU frequency under load with the same temperature targets.

3.6.2 Performance tests

When comparing the results strictly by performance then having no throttling software running gets the best results at the risk of an abrupt shutdown. Figure 36 shows this during kernel build testing as the CPU is allowed to run at maximum performance with a 100°C throttling point enforced by the CPU itself. Both LTD and LCM are on average slower by 6.05% and 4.03%, respectively, but LCM is consistently faster than LTD.

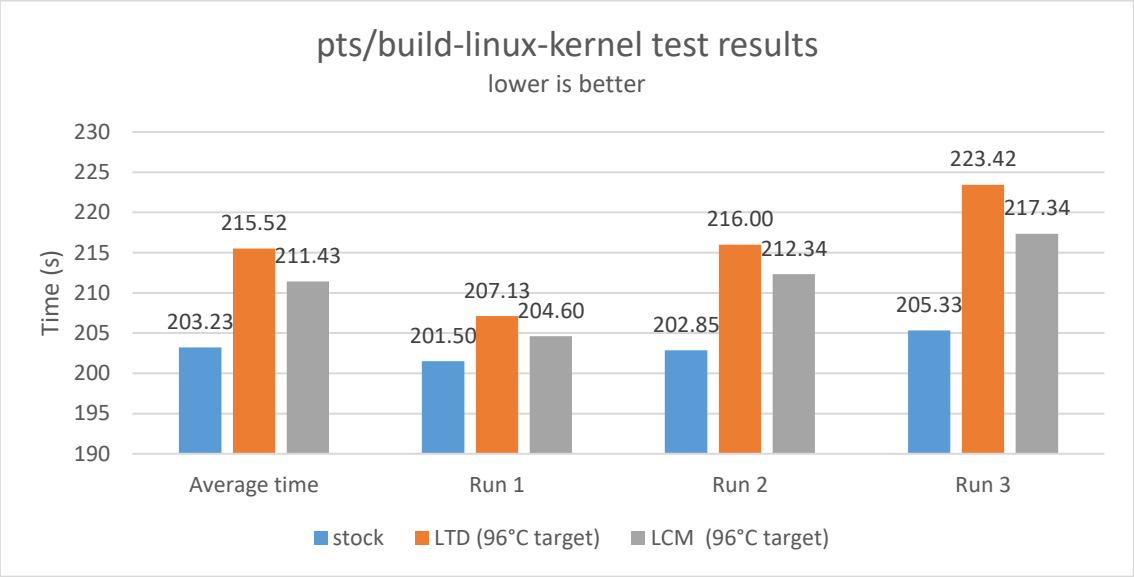

Figure 36. Comparison of Linux kernel build test results between the stock configuration, Linux Thermal Daemon and Linux CPU Manager.

LCM consistently beats LTD in terms of performance and thermal behaviour as it is more stable at a set temperature limit while also being more performant. On average, LCM is 1.9% faster than LTD.

3.6.3 Improved Linux Thermal Daemon results

Previous CPU test results show that LTD exhibits erratic behaviour and seemingly overcorrects when under a heavy load, resulting in inferior performance and less effective throttling. The main cause of this could be a default polling interval of 4 seconds which doesn't allow for LTD to have an accurate overview of the system temperatures.

A proposed solution for this is to shorten the poll interval. To test this out, the poll interval was set to 1 second and the trip point is set to 87°C. Figure 37 shows that LTD with the shorter poll interval has a positive effect on the CPU frequency as it is much more stable and less volatile.

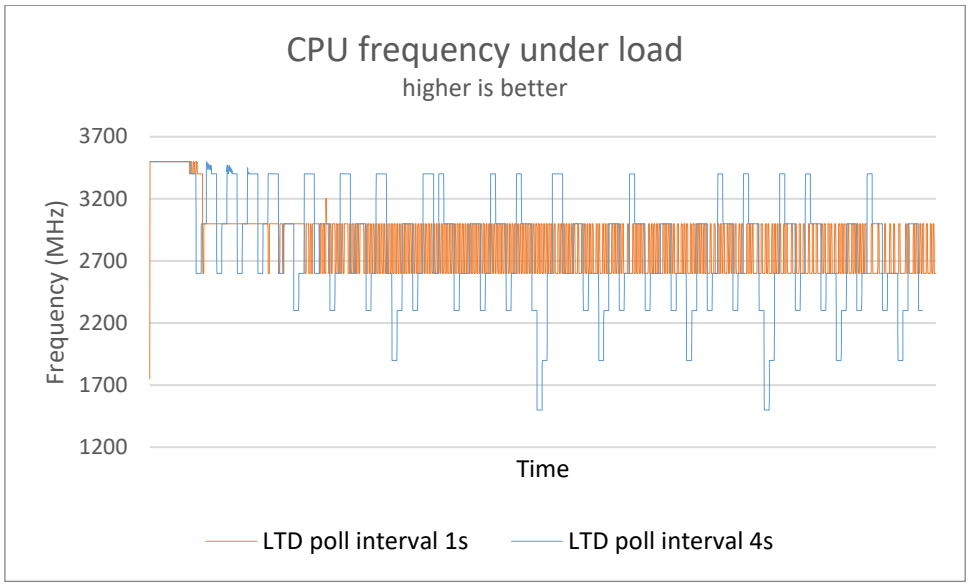

Figure 37. Comparison of CPU frequency between Linux Thermal Daemon with a shorter and longer polling interval.

As the CPU frequency is directly related to power usage and CPU package temperatures, the reduced volatility also has a positive effect on CPU temperature variance, as shown on Figure 38.

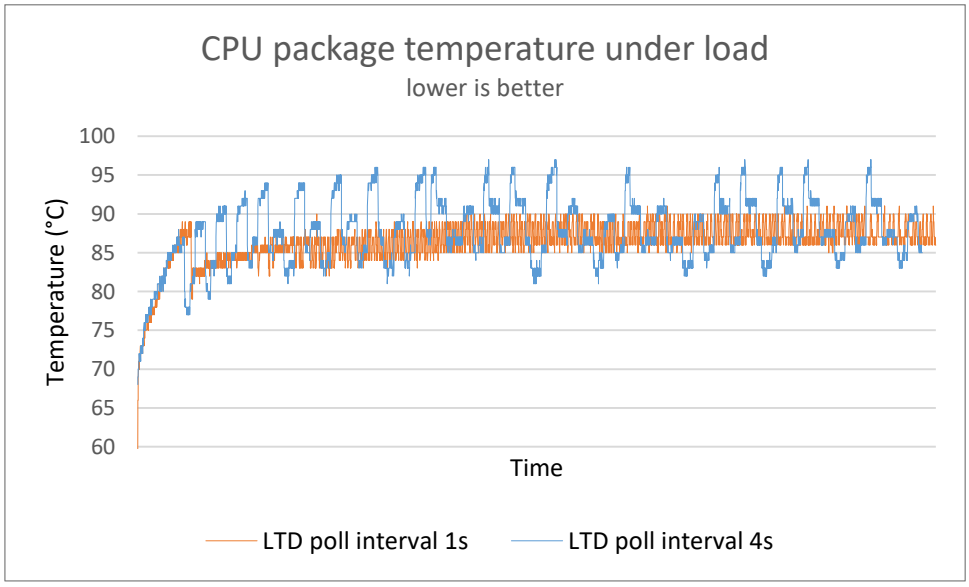

Figure 38. Comparison of CPU package temperatures under load with Linux Thermal Daemon poll interval of 1 second and 4 seconds.

While the average temperature of LTD with a shorter poll interval is at the same level as LCM and LTD with the default poll interval, the maximum temperature reading during testing is lowered from 97°C to 91°C, as visible on Figure 39.

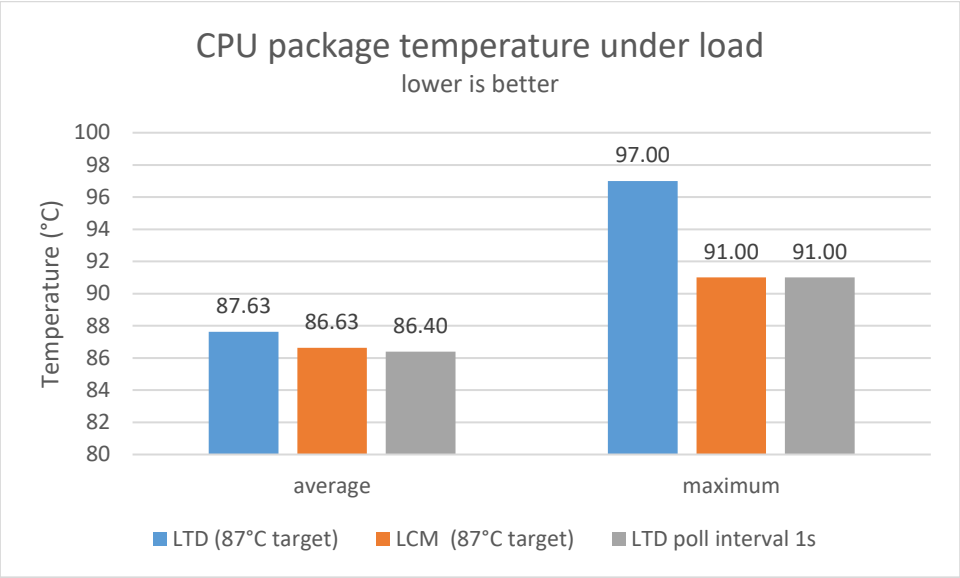

Figure 39. Comparison of average and maximum CPU package temperatures while under heavy load.

The improved LTD configuration saw great improvements, but LCM still had a higher average CPU frequency, as shown on Figure 40.

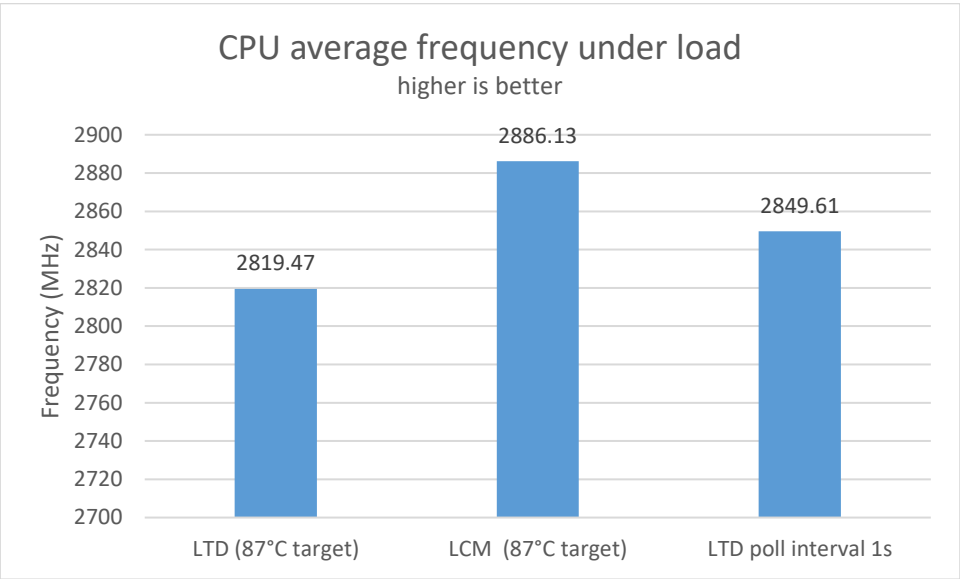

Figure 40. Comparison of average CPU frequency under a heavy load.

During testing it was observed that LTD controls the `intel_pstate` driver by reducing the `max_freq_pct` in steps of 10 while the smallest value that influences performance is 3. This comes from the step calculation in LTD which doesn't seem to calculate the number of steps correctly for this CPU. Further improvement regarding the number of steps may improve the performance of the LTD P-state cooling device implementation.

3.6.4 Linux CPU manager governor testing

Kernel build testing was also carried out using all the governors that LCM offers. As expected, the slower governors exhibited the lowest performance, as shown on Figure 41.

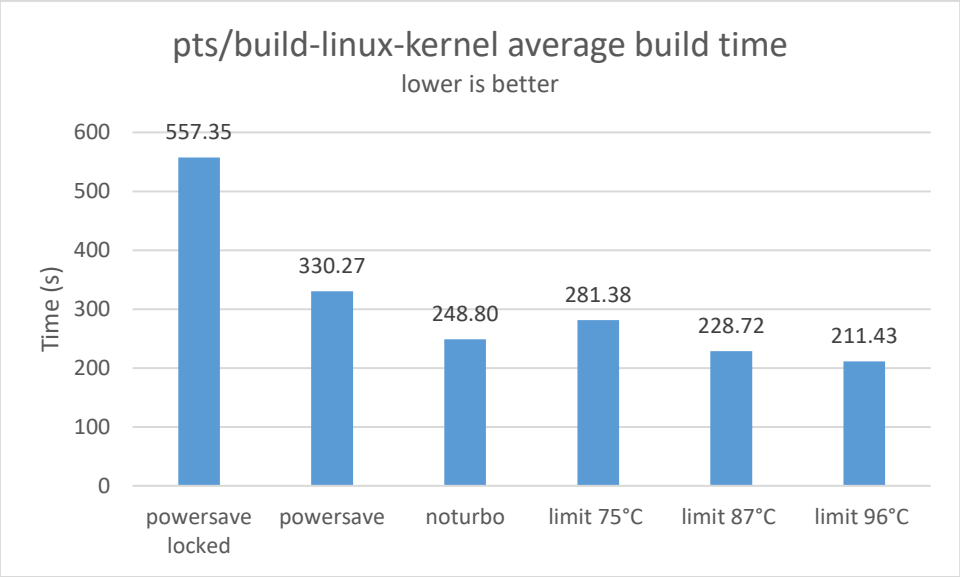

Figure 41. Comparison of Linux kernel build test results.

Figure 42 also shows a predictable pattern as lower performance governors use less power.

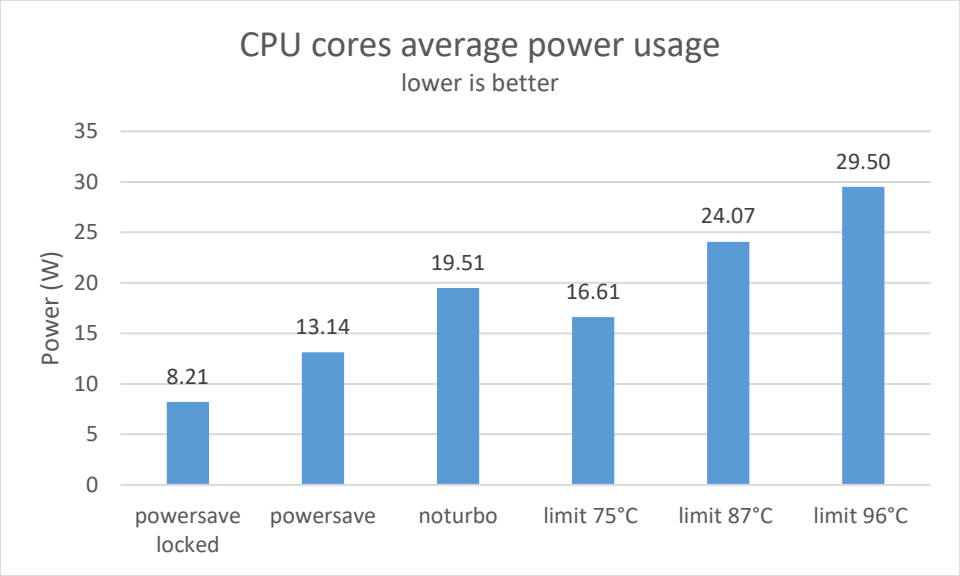

Figure 42. Comparison of CPU cores power usage between LCM governors.

Comparing the efficiency of the modes by CPU cores average power usage the most efficient mode is “powersave”, as seen on Figure 43. The task may take a longer time to finish, but the total amount of energy used by CPU cores is considerably smaller.

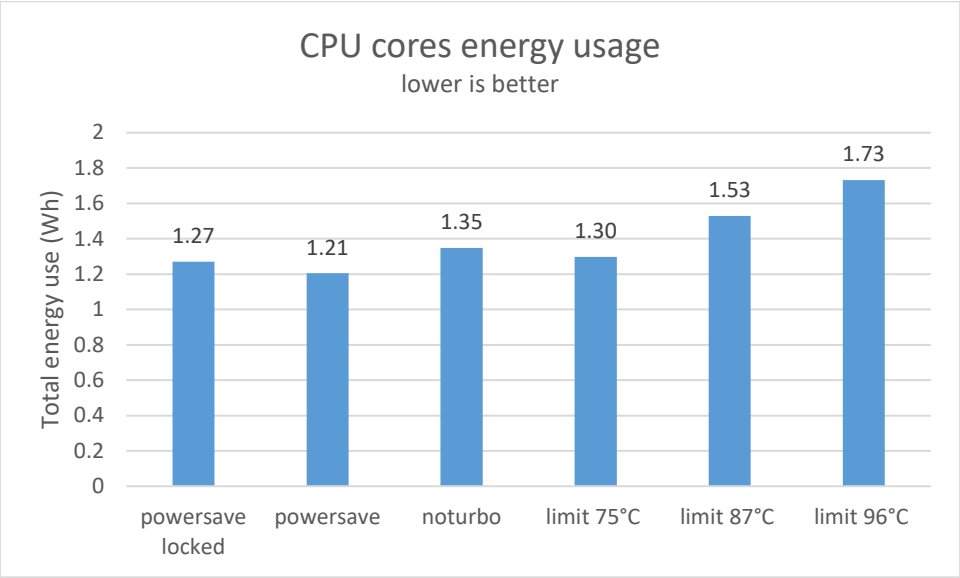

Figure 43. Comparison of total energy usage between different CPU governors during the Linux kernel build test.

However, when considering the total system power usage, the situation changes. Governors with higher clock speeds have an advantage as a 5W addition to the total system power usage shows that the "noturbo" mode is the most efficient, as shown on Figure 44. The lowest performance governor is the least efficient according to this calculation as the rest of the system components are running for a longer time.

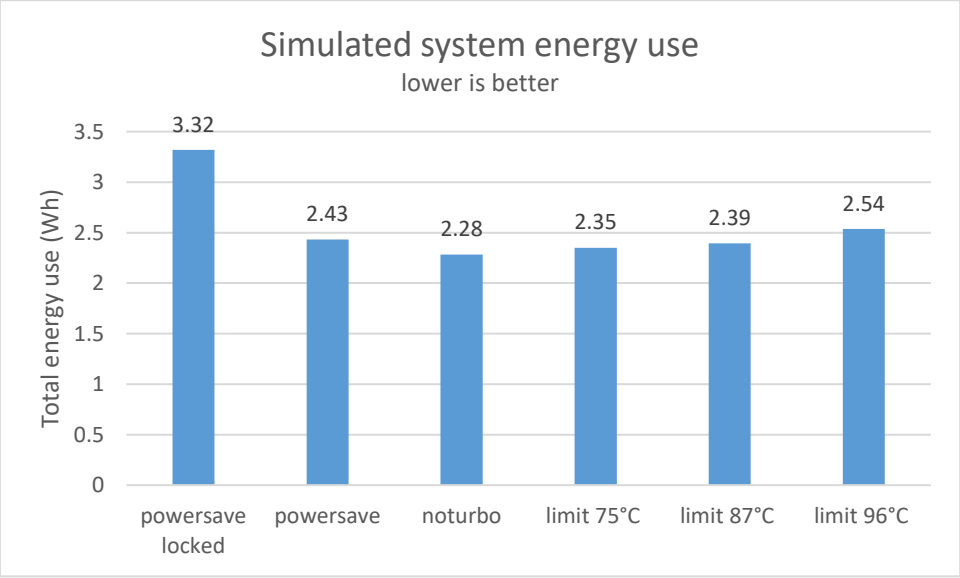

Figure 44. Comparison of total power use when taking into account a hypothetical system components power usage of 5W.

Workloads in normal use don't follow this model. Efficiency should be measured over a longer time. For example, if the computer must run for 1 hour and during that time a short but intensive workload is run it would make more sense for it to run at a lower speed.

Considering the approximate system components power usage and the CPU idle power usage we see that lower power modes are once again more efficient, as shown on Figure 45. During idling the CPU package will consume at most 4W and other system components will consume a constant rate of power over time (5-15W, depending on system components, brightness settings, network connectivity etc.). If the background system power usage is constant over all modes then the only difference is caused by the CPU power usage.

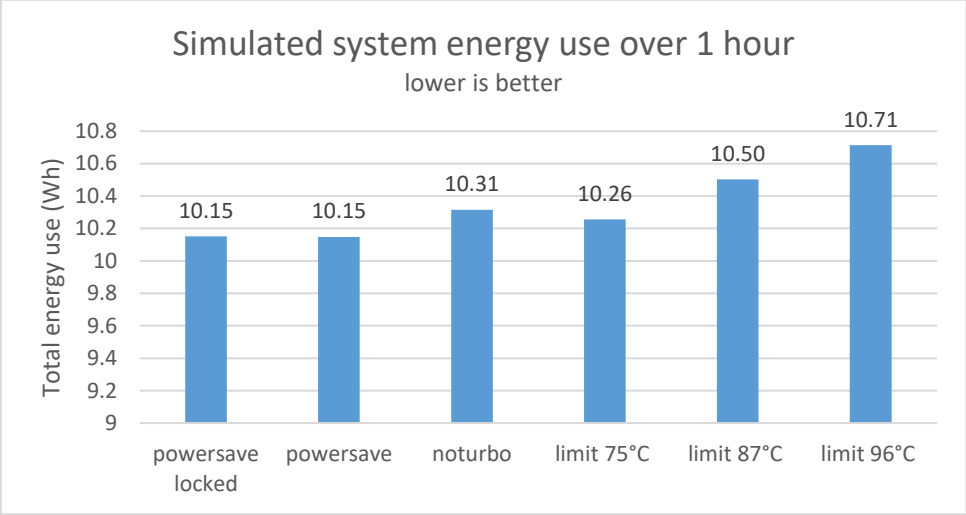

Figure 45. Comparison of total simulated system energy use over 1 hour. The workload run during this hypothetical scenario is the Linux kernel build test which takes around 3-10 minutes to finish.

3.7 Conclusions

When comparing the thermal behaviour between LTD and LCM it is clear that the LCM implementation is superior, as it provides better performance at the same temperature limit. LTD with a shorter polling period confirms this as the temperature variance was significantly reduced and the average clock speed was increased. These results confirm the findings of previous work that found that the optimal throttling method was to run the CPU at the maximum speed as long as the temperature is below a set limit.

Due to these findings an issue has been created at the Linux Thermal Daemon GitHub page and the author of this thesis will collaborate with the maintainer to improve the performance of the Linux Thermal Daemon with the proposed solution.

Power savings related findings for Linux CPU Manager indicate that running the CPU at lower speeds does increase the efficiency of the CPU. The results align with efficiency curves on CPU-s that are covered in section 3.2.1 of this thesis.

4 Summary

This thesis aimed to solve issues with poor battery life under Linux and both CPU and GPU induced overheating. To solve these issues, two services were created with an accompanying client program implementation. The services allow the user to control the performance of the CPU and GPU in order to achieve better energy efficiency, improve battery life and prevent overheating with the included pre-emptive throttling functionality.

Testing of the GPU manager solution found that for heavier graphical workloads the power usage of both CPU and GPU components can be reduced significantly. For example, the GPU power usage saw a decrease of 77.46% with a heavy 3D workload while the package power usage was only a third of the power usage when compared to running the workload without the GPU manager. Lighter workloads also had an improvement in power usage, albeit a smaller one due to the workloads being not that demanding and the GPU manager being able to influence the power usage less. GPU manager can also be used to improve efficiency by up to 31%, as seen in section 2.6.3 of this thesis. The results aligned with sources that suggested that Intel GPU-s have a known so-called “efficient frequency”.

CPU manager solution testing focused on thermal throttling behaviour and performance testing of its numerous governors. Thermal throttling tests found that the throttling behaviour and performance was noticeably better than its alternative, the Linux Thermal Daemon, exhibited. During testing the author decided to investigate the erratic behaviour of Linux Thermal Daemon and found that the problem is caused by a polling interval that is too long and a flaw in calculations which results in the applied correction being too severe. Tests with a shorter polling period confirmed the issue as the thermal throttling behaviour was much less erratic and the average CPU clock speed was slightly increased.

CPU manager solution performance testing found that limiting the CPU performance states can be used to lower CPU temperatures and power usage. Furthermore, the tests also demonstrated that the CPU efficiency can be increased as the total energy usage

during a workload was lower with the low performance governors. The results aligned with CPU efficiency curves covered in section 3.2.1 of this thesis.

Future work on the solutions created in this thesis focuses on testing on a wider range of hardware, packaging the solution into an easily installable package, the creation of GUI clients that can make use of the services and adding configuration file support. The author will also continue discussing the issues found with Linux Thermal Daemon with the project maintainer on GitHub and hopes to help solve them.

Acknowledgements

I would like to thank Kristjan Kukkur from Flex Sülearvutikeskus for providing access to a couple of laptops with Intel 8th generation CPUs for testing and analysis purposes.

References

[1] Mozilla Corporation, “Firefox Hardware Report,” 22 April 2018. [Online]. Available: https://hardware.metrics.mozilla.com/. [Accessed 22 April 2018].

[2] StatCounter, “Desktop Operating System Market Share Worldwide | StatCounter Global Stats,” StatCounter, [Online]. Available: http://gs.statcounter.com/os-market-share/desktop/worldwide/#monthly-201804-201804-bar. [Accessed 22 April 2018].

[3] NetMarketShare, “Operating System Market Share,” [Online]. Available: https://netmarketshare.com/operating-system-market-share.aspx. [Accessed 22 April 2018].

[4] L. Clark, “The whole web at maximum FPS: How WebRender gets rid of jank – Mozilla Hacks - the Web developer blog,” 10 October 2017. [Online]. Available: https://hacks.mozilla.org/2017/10/the-whole-web-at-maximum-fps-how-webrender-gets-rid-of-jank/. [Accessed 16 May 2018].

[5] Intel Corporation, “Intel® Core™ i7-3820QM Processor (8M Cache, up to 3.70 GHz) Product Specifications,” [Online]. Available: https://ark.intel.com/products/64889/Intel-Core-i7-3820QM-Processor-8M-Cache-up-to-3_70-GHz. [Accessed 16 May 2018].

[6] A. L. Shimpi and R. Smith, “Die Size and Transistor Count - The Intel Ivy Bridge (Core i7 3770K) Review,” 23 April 2012. [Online]. Available: https://www.anandtech.com/show/5771/the-intel-ivy-bridge-core-i7-3770k-review/3. [Accessed 29 April 2018].

[7] J. Mechnich, “intel-power-control: GPU power management for Intel hardware on Linux,” 20 September 2014. [Online]. Available: https://github.com/jmechnich/intel-power-control. [Accessed 26 April 2018].

[8] Arch Linux Wiki, “Intel graphics - ArchWiki,” 4 April 2018. [Online]. Available: https://wiki.archlinux.org/index.php/intel_graphics#Module-based_Powersaving_Options. [Accessed 26 April 2018].

[9] M. Larabel, “Tweaks To Extend The Battery Life Of Intel Linux Notebooks - Phoronix,” 24 August 2011. [Online]. Available: https://www.phoronix.com/scan.php?page=article&item=intel_i915_power. [Accessed 16 May 2018].

[10] B. Widawsky, “A bit on Intel GPU frequency,” 8 May 2015. [Online]. Available: https://bwidawsk.net/blog/index.php/2015/05/a-bit-on-intel-gpu-frequency/. [Accessed 16 May 2018].

[11] Free Software Foundation, Inc., “The GNU General Public License v3.0 - GNU Project - Free Software Foundation,” 29 June 2007. [Online]. Available: https://www.gnu.org/licenses/gpl-3.0.en.html. [Accessed 16 May 2018].

[12] freedesktop.org project, “Introduction to D-Bus,” 14 July 2013. [Online]. Available: https://www.freedesktop.org/wiki/IntroductionToDBus/. [Accessed 16 May 2018].

[13] P. Mochel and M. Murphy, “sysfs - _The_ filesystem for exporting kernel objects.,” 16 August 2011. [Online]. Available: https://www.kernel.org/doc/Documentation/filesystems/sysfs.txt. [Accessed 22 April 2018].

[14] Intel Corporation, “intel_gpu_frequency.c\tools - xorg/app/intel-gpu-tools - Test suite and tools for DRM/KMS drivers,” 2015. [Online]. Available: https://cgit.freedesktop.org/xorg/app/intel-gpu-tools/tree/tools/intel_gpu_frequency.c. [Accessed 16 May 2018].

[15] L. Brown, “turbostat -- show CPU frequency and C-state residency,” 18 October 2017. [Online]. Available: https://github.com/torvalds/linux/blob/master/tools/power/x86/turbostat/turbostat.c. [Accessed 16 May 2018].

[16] Intel Corporation, “Intel® Core™ i5-3320M Processor (3M Cache, up to 3.30 GHz) Product Specifications,” [Online]. Available: https://ark.intel.com/products/64896/Intel-Core-i5-3320M-Processor-3M-Cache-up-to-3_30-GHz. [Accessed 16 May 2018].

[17] Phoronix Media, “Phoronix Test Suite v7.8.0 Test Client Documentation,” 14 February 2018. [Online]. Available: https://www.phoronix-test-suite.com/documentation/phoronix-test-suite.pdf. [Accessed 16 May 2018].

[18] Phoronix Test Suite, “OpenBenchmarking.org - Java 2D Microbenchmark Test Profile,” 3 March 2018. [Online]. Available: https://openbenchmarking.org/test/pts/j2dbench. [Accessed 22 April 2018].

[19] Phoronix Test Suite, “OpenBenchmarking.org - Unigine Sanctuary Test Profile,” 27 February 2018. [Online]. Available: https://openbenchmarking.org/test/pts/unigine-sanctuary. [Accessed 22 April 2018].

[20] D. v. Vugt, “IntelQuickSyncVideo - Ubuntu Wiki,” 5 February 2018. [Online]. Available: https://wiki.ubuntu.com/IntelQuickSyncVideo. [Accessed 26 April 2018].

[21] T. Jukić, “Draw calls in a nutshell - Tonči Jukić - Medium,” 25 June 2015. [Online]. Available: https://medium.com/@toncijukic/draw-calls-in-a-nutshell-597330a85381. [Accessed 28 April 2018].

[22] C. Gough, I. Steiner and W. A. Saunders, Energy Efficient Servers - Blueprints for Data Center Optimization, Apress, 2015, pp. 36-37.

[23] P. S. Y. Deepak Rajan, “Temperature-Aware Scheduling:,” in *The Ninth International Conference on Web-Age Information Management* , Zhangjiajie Hunan, 2008.

[24] R. Rao and S. Vrudhula, “Performance Optimal Processor Throttling Under Thermal Constraints,” in *Proceedings of the 2007 international conference on Compilers, architecture, and synthesis for embedded systems*, Salzburg, 2007.

[25] L. Zhou and S. Guo, “Thermal management of ARM SoCs using Linux CPUFreq as cooling device,” *COMPUTER MODELLING & NEW TECHNOLOGIES 2014*, vol. 18, no. 12D, pp. 162-167, 2014.

[26] “Intel P-State driver,” [Online]. Available: https://www.kernel.org/doc/Documentation/cpu-freq/intel-pstate.txt. [Accessed 17 May 2018].

[27] J. P. Arjan van de Ven, “INTEL POWERCLAMP DRIVER,” [Online]. Available: https://www.kernel.org/doc/Documentation/thermal/intel_powerclamp.txt. [Accessed 17 May 2018].

[28] S. Pandruvada, “Running Average Power Limit – RAPL,” 06 June 2014. [Online]. Available: https://01.org/blogs/2014/running-average-power-limit-%E2%80%93-rapl. [Accessed 17 May 2018].

[29] C. Imes, “Ubuntu Manpage: rapl-set - set RAPL configurations,” [Online]. Available: http://manpages.ubuntu.com/manpages/bionic/man1/rapl-set.1.html. [Accessed 17 May 2018].

[30] M. Koppehel, “martin31821/cpupower: Gnome-Shell Extension for intel-pstate driver,” [Online]. Available: https://github.com/martin31821/cpupower. [Accessed 17 May 2018].

[31] M. Koppehel, “CPU Power Manager - GNOME Shell Extensions,” [Online]. Available: https://extensions.gnome.org/extension/945/cpu-power-manager/. [Accessed 17 May 2018].

[32] Intel Corporation, “Introduction to Thermal Daemon | 01.org,” [Online]. Available: https://01.org/linux-thermal-daemon/documentation/introduction-thermal-daemon. [Accessed 17 May 2018].