Repurposing Old Gaming Consoles

Nihal Umesh Konan
Master's CSE, University of California San Diego, nkonan@ucsd.edu

Sureel Shah
Master's ECE, University of California San Diego, sbs001@ucsd.edu

Colin Lemarchand
Master's CSE, University of California San Diego, clemarch@ucsd.edu

Dingye Chen
Bachelors CSE, University of California San Diego, d1chen@ucsd.edu

Chris Cha
Bachelors CSE, University of California San Diego, ccha@ucsd.edu

Eric Xiao
Bachelors CSE, University of California San Diego, erxiao@ucsd.edu

Table of Contents

ABSTRACT

Previous work has been made focused on the potential reuse of consumer electronics rather than being discarded and contributing to hazardous waste. Repurposed consumer gaming consoles are especially (should be looked at) due to their high computational power at their cost (for their time). Our project focused on the general feasibility of reusing several gaming consoles for general computational tasks, and the challenges we faced.

CONCEPTS • Repurpose game consoles • General feasibility • Computational tasks

Additional Keywords and Phrases: Consumer electronics, hazardous, consoles, computational power

1. INTRODUCTION

As demand for computing power increases, so too does demand for the limited material resources needed to manufacture the semiconductors and other electrical parts used to assemble computers.[Rockett, Angus. *The materials science of semiconductors*. Springer Science & Business Media, 2007.] Concerns over sustainable technology have prompted efforts to repurpose old hardware for new computing tasks. Our project aims to explore the feasibility and produce a proof-of-concept method for reusing gaming consoles.

Video game consoles are powerful computers: they not only have as much computing power as a standard PC, but they also come with large storage space and are able to connect to the Internet. [Zittrain, Jonathan. The Future of the Internet – And How to Stop It. (New Haven & London: Yale University Press, 2008)] However, as the gaming industry tends to release new generations of consoles on a frequent basis, older generation consoles are often abandoned and their valuable hardware is left unused. Rather than scrapping consoles for parts, we want to explore the possibility of reusing the console itself and harness its computing power. This involved using custom code provided by the console hacking community to bypass the console's security and run non-game programs. This method, if proven feasible, would introduce a more cost-efficient alternative to building computing clusters from scratch.

Our approach to develop this project is as follows:

- Install linux system and custom firmware onto the game consoles available to us, which include the PlayStation , and the Nintendo Switch.
- Leverage these resources in order to exploit and run arbitrary code on these consoles.
- Collect and validate methodologies, documentation, and guides in order to find a consistent and safe approach to "hack" the devices.

Previous work has indicated the feasibility of this approach, but many challenges still remain: while this project is building off of the work and talent of console hackers around the world, there are still many core components of this project that rely on unofficial and undocumented methodologies. Due to the scope and variable interest in specific consoles and their specific communities, there is a varying amount of information available for each of the devices we plan to use which may not be up to date. As well as this, console developers have explicitly stated their opposition to the installation of custom firmware onto their devices, also implementing security features that attempt to prevent such actions. Due to our limited range of console devices, we assigned separate teams to this task for each console, in the interest of parallelizing our work to maximize our chances of success.

In the future, we will perform extensive benchmark tests on each console to assess their potential computing power. This may include runtime, memory, and energy consumption performances. We documented mapping out the procedures of hacking the console alongside our findings so that others will be able to mimic the project in order to repurpose their own consoles.

2. RELATED WORK

1. Renée: New Life for Old Phones | IEEE Journals & Magazine -> Information about clusters and repurpose electronic devices
2. Repurposing end of life notebook computers from consumer WEEE as thin client computers – A hybrid end of life strategy for the Circular Economy in electronics - ScienceDirect -> repurposing computers as thin client comp
3. Data Centers from Discarded Cell Phones -> Creating data centres from old cell phones
4. DroidCluster: Towards Smartphone Cluster Computing -- The Streets are Paved with Potential Computer Clusters | IEEE Conference Publication | IEEE Xplore -> Smartphone clusters
5. Supercomputing with commodity CPUs: are mobile SoCs ready for HPC? -> Mobile soc performance
6. Smartphone Evolution and Reuse: Establishing a More Sustainable Model | IEEE Conference Publication -> Old smartphone reuse and repurposing

7. Demo | Proceedings of the 10th international conference on Mobile systems, applications, and services <- similar android phone cluster (university of cyprus)
8. Managing big data experiments on smartphones | SpringerLink <- using smartphone cluster for experiments with big data
Add a bunch of papers on energy efficiency, carbon cost, etc; should be enough for related works
9. GitHub - CTCaer/hekate: hekate - A GUI based Nintendo Switch Bootlo...- https://github.com/CTCaer/hekate -> How to use hecate for switch bootload
10. L4T Ubuntu Linux Install Guide - https://wiki.switchroot.org/en/Linux/Ubuntu-Install-Guide -> L4TUbuntu installation on Switch
11. RCMloader Zero - https://www.xkit.xyz/rcmloader-zero/ -? RCM loader and its usage
12. Phoronix Test Suite - Linux Testing & Benchmarking Platform, Automa... - http://www.phoronix-test-suite.com/ -> Foronix - benchmarking suite.

In [Taylor, Dave. "Need for speed: PS3 Linux!." *Linux Journal* 2007.156 (2007): 5.], the author proposes a possible means of repurposing consoles to transform them into a WebTV by installing a sterile Linux environment. The repurposed consoles are able 13to provide a Web browser and Email system, and are aiming for people who want a more lightweight solution compared to a complicated PC.

In [J. Switzer, E. Siu, S. Ramesh, R. Hu, E. Zadorian and R. Kastner, "Renée: New Life for Old Phones," in IEEE Embedded Systems Letters, doi: 10.1109/LES.2022.3147409.], Jennifer and her peers explore the feasibility of integrating used smartphones into general-purpose compute devices. By proving an 8 year old phone can still run computing tasks, they put forward an effective means to handle harmful e-wastes.

3. TECHNICAL WORK

3.1 Sony Playstation 3

While it was a difficult challenge, we were essentially able to accomplish what we set out to do with the PS3. At the time of writing, guides are few and far between, filled with broken links, outdated instructions, and contradictions. With the console becoming more and more outdated, it is unlikely that this will ever change. After weeks of research, we were able to compile information from many different sources on the internet into a comprehensive, end-to-end guide on installing our preferred Linux distribution on a compatible PS3.

3.1.1.Console Compatibility

All "fat" PS3s are compatible, some "slim" PS3s are compatible, and no "super slim" PS3s are compatible. Anything built pre-2011 is fine; i.e., CECHAxx-CECHQxx, CECH-20xxx, CECH-21xxx, and certain CECH-25xxx models. The most reliable options are the "fat" or 2000/2100 series “slim". The slim models (especially 2100-series and pre-2011 2500-series) have the smallest CPU and GPU die sizes[1] out of all the compatible consoles, and thus have lower power consumption, slightly stronger performance, and better thermals. If given the choice, it might be nicer to get one of these.

3.1.2.Jailbreaking your PS3

This video, by MrMario2011[1], explains the initial jailbreaking process quite well. After the steps in this video, the latest version of the Rebug toolbox[1] must be installed via the technique demonstrated at the end of the tutorial.

3.1.3.Setting up OtherOS++

Downgrading from EvilNat 4.88 to 3.55 Rebug CFW:

With the Rebug toolbox installed, QA flags will need to be enabled, and the 3.55 Rebug CFW[1] must be loaded onto a flash drive at /PS3/UPDATE/PS3UPDAT.PUP. Finally, the CFW "update" can be installed via the PS3 System Update menu > Install from Storage Media.

3.1.4.Repartitioning the HDD to make space for Linux

Once it finishes installing, the PS3 must be rebooted into safe mode, where the "Restore PS3 System" option can be selected. This will format, repartition, and reinstall the CFW on your PS3's hard drive, *wiping everything* in the process. It is crucial that any valuable data on the hard drive be backed up before performing this step.

3.1.5.Installing Petitboot

In order to install the bootloader, it is required to first determine whether the PS3 being used has a NAND flash or a NOR flash. The CECHAxx-CECHGxx needs the NAND version of Petitboot[1], and everything else needs the NOR version of Petitboot[1]. The bootloader file should be named dtbImage.ps3 and placed on the root of a USB drive. The Rebug toolbox file should still be on the USB drive at this stage. If it somehow isn't, it must be put back on the root, along with this script[1]. Now, the Rebug toolbox can be used to resize the VFLASH by navigating to Utilities > Resize VFLASH/NAND Regions. Once this completes, the PS3 will reboot. Finally, one navigates to Utilities > Install Petitboot in the Rebug toolbox.

3.1.6.Loading Petitboot for the first time

In the Rebug toolbox, the System > Boot OtherOS action can now be used, with the "use current" option. At this point, PS3 controllers will no longer work, and a USB keyboard is required. Once Petitboot loads, select "Exit to Shell". cd to /var/petitboot/mnt, and check which device contains the create_hdd_region.sh script. On NOR devices, it should be sda1, and on NAND devices, it will probably be sdb1. Once you have located it, run the script. It should do everything automatically at this point. This step is optional, but may be useful. Petitboot can be configured to automatically select the first option after a specified timeout, by running ps3-bl-option -O 5 (5 second timeout; this can change this to whatever is desired).

3.1.7.Installing Linux

History of PS3-supported Linux distros:

Historically, Yellow Dog Linux[1] was the distro of choice for PS3, even being officially supported by Sony. However, after Sony pulled the plug on OtherOS, and Apple stopped producing PowerPC machines, there wasn't much of a market for the distro anymore, and it was discontinued in 2012.There is a fairly popular guide[1]on the PSX-place forums describing the installation of Red Ribbon Linux, however Red Ribbon hasn't been touched since 2014, and its own official website has been down for years. The linked guide even warns people not to proceed with the installation due to the lack of support.Some of the more popular general Linux distros (e.g. Arch, Fedora, Ubuntu, etc.) have versions that reportedly work on the PS3, but they all seem to be seriously outdated and not really worth trying.

3.1.8.T2 SDE (System Development Environment)

The only distro we are aware of that is actively receiving updates for the PS3 is the T2 System Development Environment[1]. Without the experience of a Linux hobbyist, it is not easy to troubleshoot things with this distro, as it requires quite a bit of UNIX and command-line knowledge. T2 is primarily command-line based, and it does not have a graphical desktop as almost every other modern distro would out of the box. However, it is actively being developed and officially supports PowerPC64 (i.e. PS3) systems, with many platform-specific patches to compensate for the general Linux community's lack of support for less popular CPU architectures.

T2, in addition to not featuring a graphical interface as the primary mode of interaction, does not have a package manager with online repositories containing pre-built packages. Any software installed in T2 must be built from scratch, and this can make things quite difficult at times. Moreover, its documentation[1] is not exactly newbie-friendly, meaning getting up to speed with the distro and learning how to use the T2 build toolchain might be very difficult. This makes it less enticing as an option for the PS3, but seeing as there are literally no other active PS3 Linux projects that we are aware of, this is the best option. Moreover, with a bit more community support for the project, much of the difficulty involved with this OS can be reduced.

3.1.9.Performing the installation of your preferred Linux distro

At this point, a PowerPC64 ISO of any supported distro can be burned to either a DVD or USB flash drive. Then, the DVD/USB can be inserted into the PS3. If the PS3 was turned off already, it should automatically boot to Petitboot when turned on again. If it is still on from the previous instructions, run the command petitboot in the terminal to go back to the main menu. You should see your installation media listed on the menu, but before selecting it, you may need to verify that the currently selected video mode is compatible with your TV. You can press 'e' to enter the menu item editor, and make the changes you need. This

documentation explains the video modes very well. With the video mode sorted out, the installation media item can now be selected to proceed with the Linux installation. In the case of installing T2, this video[1] demonstrates the installation on the PS3, with explanations for the configuration options specific to the PS3. This video, by the creator of T2 himself [1] may also be helpful.

3.1.10.Obtaining packages for your Linux distro

Non-T2 distros will likely have their own package manager. T2, however, requires that the T2 source code be retrieved from Subversion, by running svn co https://svn.exactcode.de/t2/trunk t2-trunk in the terminal. While logged in as root, cd into the t2-trunk folder. Now, packages can be downloaded and built by using ./scripts/Emerge-Pkg. Refer to the documentation for more information on Emerge-Pkg.

3.1.11.Final Thoughts on PS3

The PS3 has large potential as a console for reuse in general purpose computing. In spite of how challenging it was to prove its worth, we believe it is absolutely worth investigating further in future works as part of a larger console cluster. With the continuing community support and ability to build cross-platform software from source, the possibilities are endless.

3.2 Nintendo Wii

The Nintendo Wii was a gaming console released November 19, 2006 in North America, selling 101.64 million units (https://www.nintendo.co.jp/ir/en/finance/hard_soft/index.html) and being the 7th best selling home gaming console of all time. At the time of release, it used an IBM PowerPC based processor, a graphics processor by ATI in a system on a chip, and 88 MB of memory shared between the CPU and GPU. We attempted to repurpose this particular console due to its large market share and the existing console hacking communities that surround it.

The Nintendo Wii was the component of our project that we deemed a failure, mainly due to a lack of software support and difficulty installing common programs. In this section, we will discuss the steps we took in attempting to reuse the Nintendo Wii as a general computing device. We will not be focusing as much on the installation process, as this is already well documented from the resources we used. As well as this, we will also be discussing potential causes of failure and what to look for in future consoles. The general homebrewing of a Nintendo Wii console has refined itself to the point where essentially every version of the console with every software configuration is capable of relatively easy soft-modding. The gold standard that is https://wii.guide/ explains the process much better than we could.After a proper homebrew installation, the next steps towards installing the Linux kernel become much less documented. Since 2004, the project has evolved from an implementation of Linux on the Nintendo GameCube, to a port to the Nintendo Wii in 2008, to a new developer and a new version of Linux (3.x) in 2017, and then to a new maintainer in 2021 (Source: https://wiibrew.org/wiki/Wii-Linux, https://github.com/DeltaResero/GC-Wii-Linux-Kernels, https://github.com/neagix/wii-linux-ngx). We are able to compile the kernel externally on another computer, and then use a Debian installer on the Wii.

In the end, we had a functional version of linux v3.12 on the Wii, complete with a version of Bash. At this point, our progress took a turn in the other direction and we became stuck in a loop of incompatibility and dependency hell. In our attempts to find suitable benchmarking solutions for all the consoles in our project, the Wii immediately became an issue. We could not get any version of GCC, Clang, or Python to compile due to many errors and issues. In attempting to find solutions to our problem, we attempted to use the PS3 as a model comparison, as both had PowerPC CPUs, both were in similar hardware generations, and both relied on open source Linux kernels for support. As well as this, we found many parallels to modern users attempting to install programs such as Python onto their PowerPC Mac products. After some research and lots of time spent attempting to fix errors, we decided that a solution would be to update the linux kernel.

Over the past decade, there have been few attempts by many members of the community to rebase the kernel to a more modern version of Linux, but these have all dwindled in interest. Any future work modernizing the software would require extensive knowledge and time focused on this component of the project. For this quarter-long project, we decided that it was infeasible to make any meaningful progress towards our own updated kernel, as we lacked the understanding and domain knowledge required for such a project. We had even attempted to contact the original creator of the project, the user who updated the kernel to Linux 3.x, and the current maintainer, to no avail (at this time). At this point, we considered the time spent on the project, how much more time it would take to get it to a benchmarkable state, and decided that it was in our best interest to put our efforts into other parts of the project. Thus, we decided to cut our losses and cease progress on the Wii.

Our exploration of repurposing game consoles for general computing use relies heavily on community support and resources. However, the particular community for hacking the Wii has been seeing little to no activity in recent years. Given that the Wii is completely out of production, we do not expect this community to make a resurgence either. Most of the community around hacking gaming consoles has moved onto newer consoles. We also find to our disadvantage that the Wii's potential as a general purpose computing device was not pursued by Nintendo. For example, Sony had produced OtherOS for the Playstation 3, and this feature allowed user-installed software, such as Linux, to run on the Playstation 3 hardware. Even though this feature had been removed, this enabled online communities to reverse engineer the feature. Nintendo has made no such effort for their consoles, and has actively attempted to restrict similar exploits ever since, such as removing the internet browser from newer consoles. Furthermore, we speculate that the Wii will perform poorly compared to the other, easier-to-hack consoles in our project. The Wii, being an older console, is not as powerful compared to the Playstation 3 or the Switch. In fact, we learned that the Wii is based on the PowerPC CPU, nearly identical to the CPU in the Gamecube. Given its expected poor performance and the many roadblocks we faced, we saw little motivation to continue our efforts to hack the Wii.

While we faced many issues with the Wii, we believe that it gave us a better framework for considering future consoles for this project. Before this point, we had a general idea of what was capable of being turned into a general computing machine, and we wouldn't be able to effectively turn a Nintendo GameBoy into a matrix multiplying machine. The issues of the Wii highlight a potential low bar at which previous consoles should be avoided, and future consoles should be more considered. However, the failures of the Wii were not only based on performance, but also community support. We feel the need to highlight how important a strong and dedicated community is towards these types of projects. Without an existing foundation, this type of project would be nearly impossible to implement. Thus, the popularity of the console, both in its original releases and in the console hacking scene, plays a major factor in what potential community already exists around it. While this part of the project was ultimately a failure, we definitely spent a lot of time ideating and learned many things. Many of the notes and suggestions we found were from random forum posts ranging from 2004 to 2021, and we found many interesting internet rabbit holes that were as baffling as they were intriguing. Even if we decided to stop work on the Wii, there is still potential in this component of the project, and future progress can still be made. However, from a usability and sustainability standpoint, we place the Wii as unsuited for this project.

3.3 Nintendo Switch

Figure1: The Nintendo Switch with Ubuntu 18.0.4

The Nintendo Switch was installed with a Custom Firmware(CFW). Currently, two hardware revisions of the Switch exist. Any Switch bought or manufactured before the middle of 2018 has a bootrom bug that allows us to run code regardless of the firmware version on the Switch. When Nintendo updates the system, however, CFW will usually need an update to account for it. This bug cannot be fixed by Nintendo once the console leaves the factory, unless the console is sent in for repairs. This means that all current and future firmwares will be able to launch CFW through this exploit on the old hardware revision. Any console purchased after approximately August 2018 is likely to be patched. This includes the latest units on shelves, referred to as 'red box' or 'Mariko'.

Mariko is hardware patched, but may come on a vulnerable firmware. Currently, the only way to know if your Switch is hackable is by trying to send the payload in RCM. Even with this exploit fixed, many Switches on 8.0.1 and below will be hackable to some degree in the future.

To launch CFW through the exploit, the Switch needs to be in "Recovery Mode"(RCM). The easiest way to enter RCM is by grounding pin 10 in the right joycon rail and holding VOL+ on boot. Several methods and designs to do this exist. Once the Switch is in RCM it needs to be connected to either a computer, phone or loader dongle to send the exploit and the payload. This procedure needs to happen every time the Switch boots from a completely "off" state, otherwise the Switch will boot into the stock firmware. Fusée Gelée is the cold boot vulnerability that is exploited for this.

3.3.1.RCM Loader & its features

- This is a custom hardware dongle that people can use to install custom firmware onto their Nintendo Switch.
- Compatible with all Nintendo Switch consoles manufactured before June 2018
- It is designed to transfer payload.bin files to your console.
- You can transfer Atmosphere, ReiNX, and Hekate payload files with one click change.
- It is a rechargeable device.
- You can send payload 1000 times with a 1-hour charge.
- Easy to use, it will automatically be recognized as a memory card when you connect it to a computer or smartphone.
- Compatibility with Windows, macOS, Linux, and Android operating systems.

3.3.2.Steps to inject code with an RCM Loader

- Step 1 – Power Off Your Nintendo Switch You need to power off your Nintendo Switch first. Press and hold the Power button for 3 seconds to open the Power Menu and select the Power Off option.
- Step 2 – Insert The RCM Jig Remove the right Joycon from the console. Then, insert the RCM jig that comes with the dongle into the groove.
- Step 3 – Press Volume Up + Power Button At this point, press the volume up button on the Switch, followed by the power button.
- Step 4 – Connect RCM Dongle To Charging Port Once that’s done, connect the RCM dongle to the USB-C charging port at the bottom of the Switch.
- Step 5 – Remove The RCM Jig You may notice some text flashing on the Nintendo Switch’s screen. When you do, you can remove the RCM jig and the dongle and reattach your right Joycon controller.
-

3.3.3.Hekate - Nyx

Hekate from Nyx is a Custom Graphical Nintendo Switch bootloader with firmware patcher and many more tools. We used it to install our chosen CFW on the Switch. Hekate features:

- Fully Configurable and Graphical with Touchscreen and Joycon input support
- Launcher Style, Background and Color Themes
- HOS (Switch OS) Bootloader -- For CFW Sys/Emu, OFW Sys and Stock Sys
- Android & Linux Bootloader
- Payload Launcher
- eMMC/emuMMC Backup/Restore Tools
- SD Card Partition Manager -- Prepares and formats
- SD Card for any combo of HOS (Sys/emuMMC), Android and Linux
- emuMMC Creation & Manager -- Can also migrate and fix existing emuMMC
- Switch Android & Linux flasher
- USB Mass Storage (UMS) for SD/eMMC/emuMMC -- Converts Switch into a SD Card Reader USB Gamepad -- Converts Switch with Joycon into a USB HID Gamepad Hardware and Peripherals info (SoC, Fuses, RAM, Display, Touch, eMMC, SD, Battery, PSU, Charger)
- Many other tools like Archive Bit Fixer, Touch Calibration, SD/eMMC Benchmark, AutoRCM enabler and more

3.3.4.L4T Ubuntu

L4T Ubuntu Bionic (18.04) is a version of Linux based on nvidia's linux for tegra project. It uses a different kernel compared to previous releases which allows it to use features not yet in mainline. Such as audio, docking support, vulkan, and CUDA.

NOTE: The Nintendo Switch is an ARM64 Device and will NOT run x86_64/x86 software natively. This means NO Steam, Play-On-Linux, Wine x86, etc natively.
Features:

- Full USB Power Delivery/On-The-Go/HDMI/Displayport Docks
- Bluetooth/Wifi (both controllers and audio)
- Full Joycon Support (Excluding NFC, and IRcamera)
- Nvidia Tegra GPU drivers - Vulkan, OpenGL, OpenGLES, and CUDA
- Audio - Headphones, Speakers, HDMI when docked, and Microphone (version 3.4.2+)
- Touchscreen
- Handheld (720p @ 60) and Docked Display (up to 4K @ 30 or 1440p @ 60)
- CPU frequency scaling - Frequency depends on the Nvidia Applet Power Mode, automatic Console and Handheld Profiles
- Automatic Overclocking through the Nvidia Indicator Applet (select an OC mode)
- Full sdcard speed
- Hardware video acceleration when using the new SMPV player app (smpv), ffmpeg-l4t, mpv-l4t, and the built in videos/ totem app (means you can watch videos without battery draining massively)
- LP0 (lowest power) Sleep mode
- IMU/Ambiant Light sensor support reboot2payload
- Minerva Memory Training
- Support for Moonlight-QT (stream games/desktop from your Nvidia (nvidia gamestream) or AMD (sunshine moonlight host) GPU to your Switch)
- Initramfs updates (fix partition resize, add boot logo, and error screens)

3.3.5.Installation

Download the image (all users use Standard) from the downloads section. Partition with Hekate (Tools -> eMMC SD Paritions USB -> Partition SD Card), make sure to leave enough room on fat32 to extract chosen image. (this will delete all data on the SD card, Hekate will attempt to backup 1GB of data from the fat32. Make sure to backup your emuMMC, Android data, and anything else you might want to save yourself) Extract 7z to sd fat32 partition, now labeled SWITCHSD. You can use Hekate UMS to mount your SD to your PC. Flash image in partition manager in hekate to linux partition. If you have joycons: In Hekate, go to NYX tools, and dump joycon pairing data, with joycons connected to console, after being paired in hos (Home -> Nyx Options -> Dump Joy-Con BT). Launch Ubuntu through Hekate (Home -> More Configs -> Ubuntu). Wait for around 2-4 minutes while it initially loads. Follow the initial setup and you are done. You now have fully featured Ubuntu on your switch!

4. MILESTONE

Deliverable 1: Hacking the console		
Milestone 1	Install custom framework on three consoles.	Due April 29 (Week 5)
Milestone 2	Install Linux system on three consoles.	Due May 6 (Week 6)
Milestone 3	Run and compile arbitrary code on the Linux system.	Due May 13 (Week 7)
Deliverable 2: Performance metrics		
Milestone 4	Run benchmarks to compare consoles' performance	Due May 20 (Week 8)
Deliverable 3: Course work		
Milestone 5	Milestone report	Due May 17 (Week 8)
Milestone 6	Develop a web presence	Due May 26 (Week 9)
Milestone 7	Finish the report and the video	Due June 6 (Week 10)

Table 1: Milestones

4.1.Deliverable 1: Hacking the console

We were able to hack all the three consoles on time under the time frame, but however we weren't able to install linux on the Nintendo Wii due to certain technical restrictions as mentioned earlier.

4.2.Deliverable 2: Performance metrics

We were able to create benchmarks for the PS3 & the Switch, where we ran multiple benchmarks such as matrix multiplication, image resize, KNN, Fibonacci and Pybench and we were able to get the results for both the consoles.

4.3.Deliverable 3: Coursework

We were on time able to complete the coursework requirements/assignments successfully such as the report, web presence and the video.

4.4.What worked really well for us?

We were on track able to complete the project for two of the consoles and were able to create benchmarks for both. We were able to quantify the CPU/GPU performance of the same.

4.5.What didn't work for us?

Due to a few restrictions we were not able to install linux and run the benchmarks for the nintendo wii. We also didn't have sufficient time to evaluate the energy efficiency and carbon cost estimates for the project and we plan to do the same in the future.

5. RESULTS

5.1 PS3 Benchmarks:

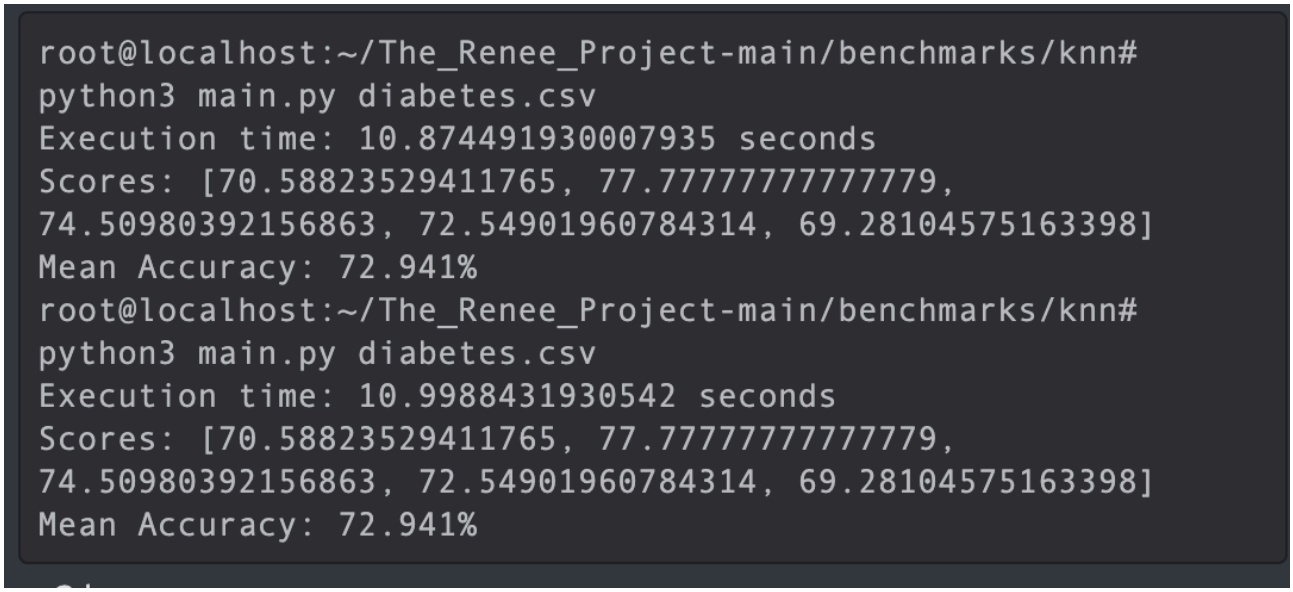

Figure 2: KNN Benchmark

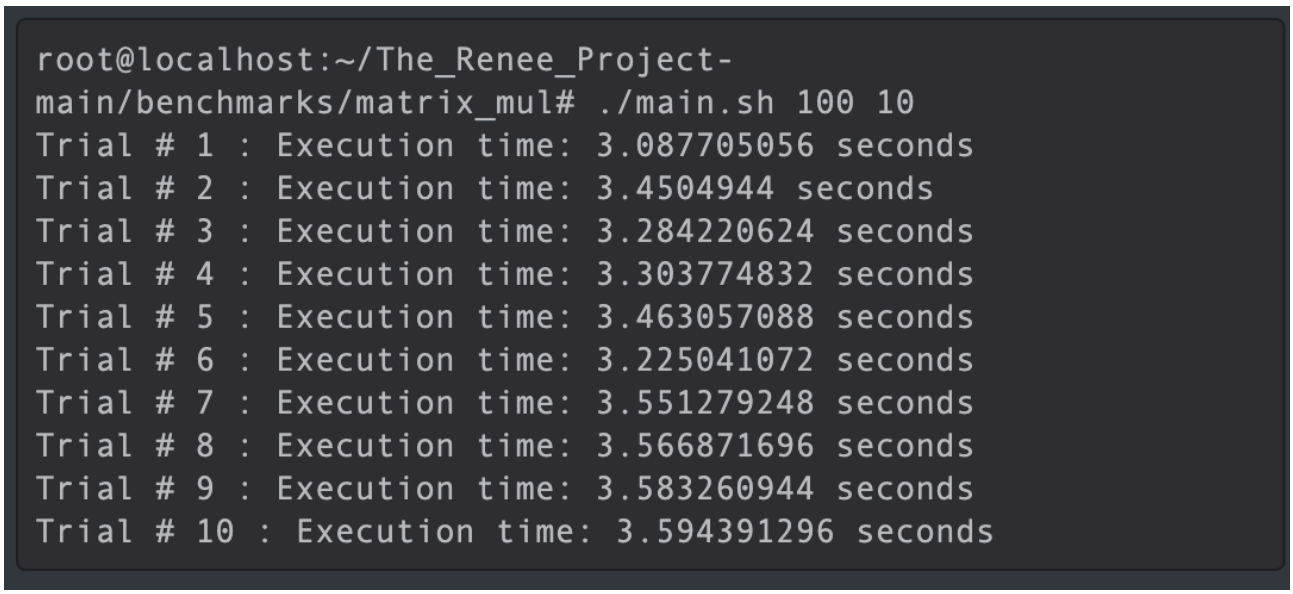

Figure 3: Matrix Multiplication(100x100):

5.2 Switch Benchmarks:

Figure 4: KNN Benchmark

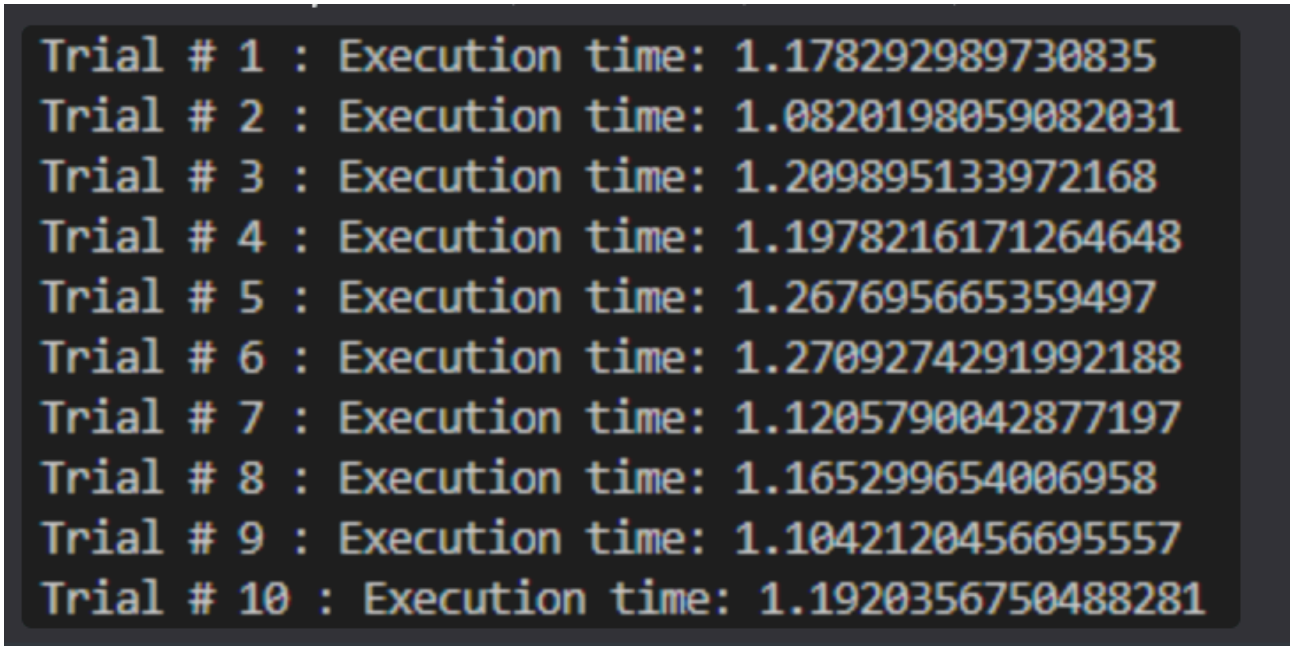

Figure 5: Matrix Multiplication(100x100):

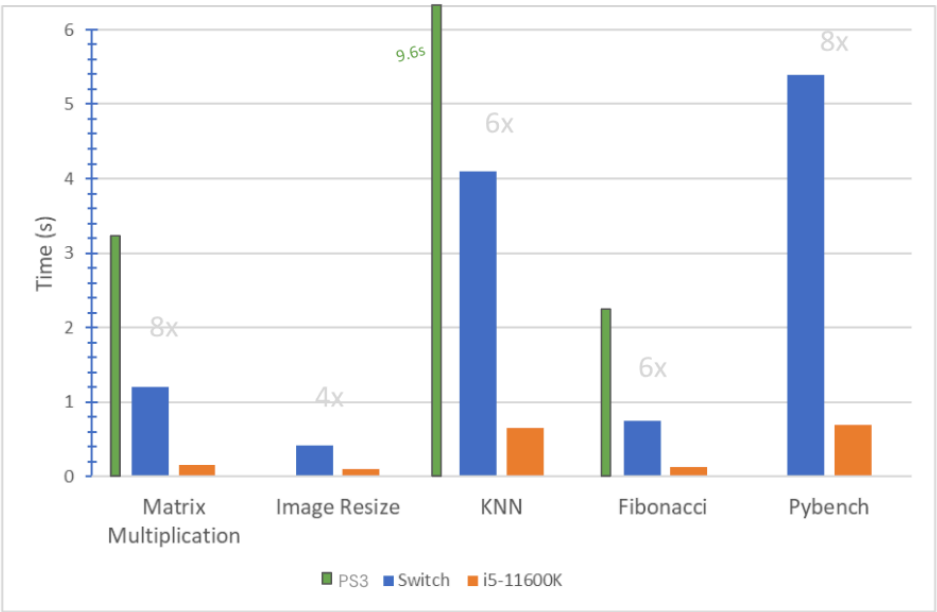

Figure 6: Benchmark results on PS3, Switch and i5

	Matrix multiplication	Image resize	KNN	Fibonacci	Pybench
PS3	3.20s	NA	9.6s	2.20s	NA
Switch	1.120s	0.4s	4.1s	0.6s	5.25s
i5	0.120s	0.1s	0.75s	0.095s	0.6s

Table 2: Execution time for each benchmarks in the consoles

6. CONCLUSION

6.1. Project Overview

Gaming consoles are highly specialized devices, with built-in security measures to prevent them from running programs other than games. However, as the gaming industry tends to release new generations of consoles on a frequent basis, older generation consoles are often abandoned and their valuable hardware is left unused. Rather than scrapping consoles for parts, we explored the possibility of reusing the console itself and harness its computing power. This involved using custom code provided by the console hacking community to bypass the console's security and run non-game programs. This method, if proven feasible for production, would introduce a more cost-efficient alternative to building computing clusters from scratch.

6.2.Project Approach

We successfully installed custom firmware onto the game consoles available to us, which include the PlayStation 3 and the Nintendo Switch. These consoles already have communities centered around console hacking, and we leveraged these resources in order to exploit and run arbitrary code on these consoles. We collected and validated methodologies, documentation, and guides in order to find a consistent and safe approach to "hack" the devices.

Due to our limited range of console devices, we installed custom firmware on a Nintendo Wii emulator in order to explore the feasibility of our methods across consoles. We assigned separate teams to this task for each console, in the interest of parallelizing our work to maximize our chances of success. We documented issues that occurred and reasons for such failures. The data from the benchmarks is compiled and analyzed for our final report.

6.3.Goals/Minimum Viable Product/Future work

At a high level, our minimum viable product consists of two of our consoles running custom software that allows us to perform basic computing tasks. In particular, we bypassed the consoles' security mechanisms via various software exploits (softmods), and installed a custom firmware allowing for freer access to the consoles' hardware than intended by the original manufacturers. With this modified firmware, we installed a Linux kernel of some sort, which enables the execution of arbitrary code just as on a typical personal computer. We required this kernel to support a C compiler such as GCC in order to recompile various programs for the unique, now-obsolete PowerPC architectures of the Playstation 3 and Nintendo Wii. The Nintendo Switch is much more modern, and unlike our other consoles, its ARM architecture is widely supported, so some benchmarking tools ran natively on the Switch.

The benchmarks of the consoles were compared to the performance to that of a personal computer and/or other consoles. It provided a quantifiable metric to compare some of the platforms. In addition to computational performance, our project will take into account the power consumption of the consoles in the near future. It may turn out that some consoles perform so poorly that the amount of power they consume is not worth the work they accomplish. If it is more environmentally friendly to recycle the consoles' electrical components than to repurpose them as computers, then we would suggest that this be done. We intend to deliver some level of quantitative analysis of how cost-effective and eco-friendly running these consoles would be, especially on larger scales.

REFERENCES

1. Jennifer Switzer, Eric Siu, Subhash Ramesh, Ruohan Hu, Emanoel Zadorian, Ryan Kastner. Renée: New Life for Old Phones, IEEE Embedded Systems Letters, 28 January 2022,
2. Rockett, Angus. *The materials science of semiconductors*. Springer Science & Business Media, 2007
3. Zittrain, Jonathan. The Future of the Internet – And How to Stop It. New Haven & London: Yale University Press, 2008
4. Taylor, Dave. "Need for speed: PS3 Linux!." *Linux Journal* 2007.156 (2007):

CITATIONS

[1] https://game-consoles.super.site/