Analysis of Android Custom Kernels

Surya M[1], Rajendra M [2]

[1]Dept. of CSE, Atria Institute of Technology, Karnataka, India
[2]Asst. Professor, Dept. of CSE, Atria Institute of Technology, Karnataka, India

---***---

Abstract - *Android is the most popular mobile platform which is based on the Linux kernel. But their devices don't age well compared to the competition. Smartphones phones ship with a stock kernel configured to provide balanced performance. Smartphone enthusiasts have come up with several techniques to boost the efficiency of mobile systems to make smartphones more effective. One such method is optimizing the kernel which can potentially accelerate the system performance. A community of smartphone enthusiasts modify the kernel from the kernel sources and release a more performance oriented custom kernel. In this paper we analyze and improve the performance of an Android phone using these custom kernels. We conduct experiments on POCO F1, with four different kernels, namely Lineage stock kernel, Illusion v5 kernel, Lawrun v10 kernel and Arter97 kernel. Results show that system performance improves by 10% to 30%. Specifically, we measure overall performance improvements of key Android system components such as multithreading and task scheduling, Binder, and storage and file systems using tools like Antutu and Geekbench. The custom kernels usually overclock the CPU and GP which showed significant performance improvements and stable functioning with a minor increase in device temperature and reduced battery life.*

Key Words: **Android, Linux, Kernels, Performance, Lineage, Illusion, Lawrun, Arter97, Antutu, Geekbench.**

1. INTRODUCTION

Since the release of Apple's first iPhone in 2007, the smartphone market has grown very rapidly. The most popular mobile operating system is the Android introduced by Google. The Android OS is based on the Linux kernel and was first released in 2008. It is an open-source platform backed by Google and served by major hardware and software developers. Android's worldwide market share has risen explosively. Because of the open nature of Android, many third-party applications are currently available. But the main problem of android ecosystem is that the phones don't get updates and there is a lack of focus on optimizing the OS and apps for older devices. Also considering the fact that old phones with the stock kernel are configured to run under balanced more. To counter this smartphone lovers are enthusiastic about 'tuning up' their mobile phones and increasing their usability. Some want to improve performance by speeding up the phone's processor.

Another method is by making more memory available to run apps faster. Others want to upgrade older models with newer versions of the operating system using custom roms. A specific set of smartphone/tech enthusiasts have formed a community in this niche market of getting maximum performance out of these smartphones. In this paper we analyze and benchmark custom kernels for POCO F1 and how the kernel performance improves the overall usage and efficiency of the device. This brings about more lifetime and usability of the device.

2. ANDROID ARCHITECTURE

From an architecture point of view, the Android operating system is divided into four layers: the kernel layer, libraries and runtime layer, applications framework layer, and applications layer. Android kernel is a modified version of Linux kernel, which is updated from time to time with different versions of Android. The libraries provide support for graphics, media capabilities, and data storage. Android runtime, embedded in the libraries layer, contains the Dalvik virtual machine to power the applications .As a replacement of Dalvik, Android introduced its new Android RunTime (ART) with ahead-of-time (AOT) compilation, which improves performance. All applications use the applications framework API for accessing the lowest level of the architecture. The layers consist of multiple program modules such that the operating system, middle-ware and other central applications. Each upper layer avails the services of its lower layer.

3. KERNEL LAYER

The kernel used by Android Operating System is a modified version of Linux Kernel to carry out some special requirements of the platform. Linux was chosen since it is open source, and has verified pathway evidence. Drivers are needed to be rewritten in various cases. Linux kernel provides mechanisms for networking, drivers, virtual memory management and power management. It works at ease with hardware and also holds all the necessary

hardware drivers. Drivers control and are in touch with hardware. For example, all devices contain Bluetooth hardware, so the kernel must also provide a Bluetooth driver for Bluetooth hardware to communicate with. Linux kernel is used for process management, memory management, networking, security settings etc.

4. CUSTOM KERNELS

The kernel which comes with Stock ROM is stable and lives up to what the OEM promises. However, as Android is all about customization, after Custom ROMs, Custom Kernels are the go-to choice among the users. Android is a famous operating system that features a lot of custom kernels out there for almost every phone nowadays. Custom Kernels not only offer security updates, but also various improvements over the Stock Kernel.

However, the kernel has complete control over the system. That means that not only a Custom Kernel can enhance your experience but can also damage your system if tinkered wrongly.

The device manufactures release the kernel sources for the device on top of which customization or modifications are done and compiled to release a custom kernel.

Custom kernels are better in more than a few ways with android. You can tweak various features like:

1. CPU, GPU and MEMORY overclocking.
2. How busy the CPU should be before it enables extra cores that it normally has disabled to save battery.
3. The CPU governor (which determines how quickly it tends to ramp up the frequency or not)
4. Change the voltage of the CPU during all possible frequencies.
5. Enable USB fast charge
6. Configure the I/O scheduler.
7. Overclock Display refresh rates.

There are different custom kernels, giving you a plethora of choices to pick from. From a high-performance, overclocked gaming kernel, or an under-clocked battery booster, or go with a fair share of both. If you can't decide, you can very well go ahead and build your own custom kernel with the features you like if the kernel sources are available.

Kernels used for analyzing system performance are:

Illusion: Compiled with AOSP clang, support for wireguard and westwood as default TCP congestion control algorithm and some standard governors.

Arter 97: Compiled with latest toolchains, Permissive SELinux status while passing SafetyNet and still keeping optimizations from Stock Kernel are some of the highlighted features of this kernel
LawRun: Compiled with android gcc, Permissive SELinux with a variety of CPU and I/O governors. With support for features in Kali Nethunter.

5. EXPERIMENT

5.1 TOOLS NEEDED

The tools needed for flashing/installing custom kernel, root access and benchmarking are as follows:

1. UBUNTU OS: A Debian Linux distribution.
2. FASTBOOT: Android flashing and booting utility.
3. TWRP: TeamWin Recovery Project is a touch based open source recovery image for android devices. Used to flash new software images in various android partitions and many other things such as formatting a partition, debugging etc.
4. MAGISK: An open source systemless rooting solution for Android based on phh's SuperUser.
5. FRANCO KERNEL MANAGER: A kernel management toolbox.
6. ANTUTU: An all-in-one benchmark tool designed to run tests on a device's CPU, GPU, memory and storage.
7. GEEKBENCH: A cross-platform processor benchmark tool.

The device used here is POCO F1 with Snapdragon 845(10nm) octa-core (4x2.8 GHz Kryo 385 Gold & 4x1.8 GHz Kryo 385 Silver) with Adreno 630 GPU and UFS 2.1 Memory.

5.2 INSTALLING CUSTOM KERNEL

First, the bootloader of the device has to be unlocked. This enables us to use a custom recovery image and install the kernels and also gain root access.

[Step1: Preparation]

1. Install fastboot on Ubuntu/Windows.
2. Download TWRP recovery image, kernel zip and Magisk zip
3. On the phone enable developer options by tapping build number 5 times.
4. In developer options enable OEM UNLOCKING.
5. Reboot the phone into FASTBOOT MODE by pressing volume up + power button.
6. Connect the phone in fastboot mode to the pc.
7. Run the command to unlock the bootloader.

```
# fastboot oem unlock
```

8. Reboot into fastboot mode.

[Step2: Installing Custom recovery]

1. Check if the device is connected using the fastboot command on the terminal.

```
# fastboot devices
```

2. With the recovery file in the current working directory install TWRP image.

   # fastboot flash recovery twrp.img

[Step3: Installing Kernel and Magisk]

1. Using the install option in twrp select the kernel zip and then Magisk zip
2. Use the swipe to install option and wait for both the zips to finish.
3. Reboot the phone.

[Step4: Kernel Manager]

1. Install Kernel Manager of any preference. We are using the Franco kernel Manager from Play Store.

5.3 ANALYSIS PROCEDURE

We use Franco kernel manager to monitor system properties when idle and active. Franco kernel has live monitor mode to visualize the system status.

Steps followed:

1. Check Franco kernel manager to check kernel status.
2. Run Antutu and Antutu 3D on the device.
3. Wait the temperature of the device to come back to normal.
4. Run GeekBench.
5. Log the scores.

Antutu benchmarks:

1. CPU
2. GPU
3. MEMORY
4. UX

Geekbench Benchmarks:

1. Single-Core performance
2. Multi-Core performance

Governor set to schedutil for all the tests and all other kernel parameters set to the default values of the custom kernel. Tests were done with stable room temperature.

5.4 RESULT

The stock kernel has the lowest performance scores compared to the custom kernels as expected.

The increase in the clock speed of the GPU and CPU with different kernel sources saw a big boost in performance. These scores are specific to the device and the chipset used along with the kernel.

Table -1: Antutu Benchmark scores for POCO F1

POCO F1 - Antutu Benchmark					
Kernel Name	CPU Score	GPU Score	Memory Score	UX Score	Overall Score
Stock	51767	69977	40908	30402	193054
Illusion	96607	123418	49159	56029	325213
Arter97	98473	127027	50508	58398	334406
LawRun	103819	140131	52613	56355	352918

Chart -1: Antutu Benchmark scores for POCO F1

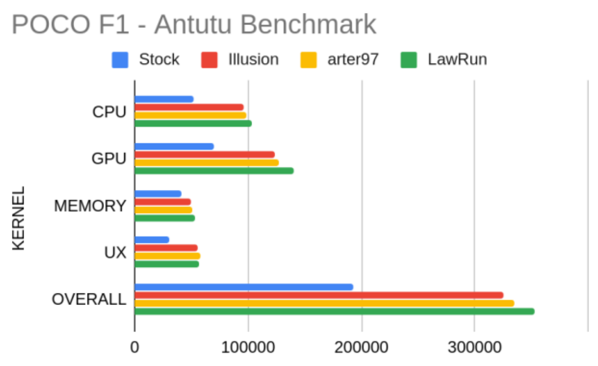
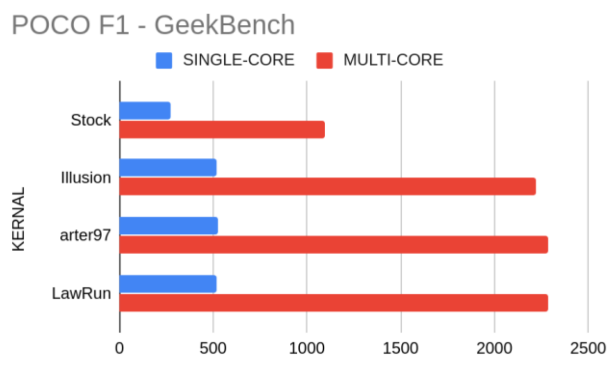


Table -2: Geekbench Scores for POCO F1

POCO F1 - GeekBench		
Kernel Name	Single Core	Multi Core
Stock	276	1098
Illusion	522	2223
Arter97	523	2285
LawRun	520	2290

Chart -2: Geekbench Scores for POCO F1



6. CONCLUSION

From the results it's very clear that stock kernels are not focused for high performance and lean towards efficiency and limited features. The custom kernels unlock a lot of features which are made available only for some latest and flagship counterparts like overclocking the display refresh rate considering the fact that the hardware supports them. For performance evaluation, we have collected benchmarks for different kernels for key Android system components such as multithreading and task scheduling, Binder, and storage and file system. Experimental results show that the average system performance speedups 10-30%.

REFERENCES

[1] P. Yuan, Y. Guo, X. Chen and H. Mei, "Device-Specific Linux Kernel Optimization for Android Smartphones," 2018 6th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud), Bamberg, 2018, pp. 65-72.

[2] L. Corral, A. B. Georgiev, A. Janes and S. Kofler, "Energy-Aware Performance Evaluation of Android Custom Kernels," 2015 IEEE/ACM 4th International Workshop on Green and Sustainable Software, Florence, 2015, pp. 1-7.

[3] Joo, B.-G & Kim, S.-M. (2012). A user's experience in optimizing smartphone performance using overclocking and memory cleaning techniques. 6. 127-138.

[4] Google Developers (2020) Android Platform, [online]. Available at: https://developer.android.com/guide/platform

[5] akhilnarang (2020) Illusion Kernel, [online]. Available at: https://forum.xda-developers.com/poco-f1/development/derp-kernel-v1-0-gcc-7-4-9-133-t3856615

[6] arter97 (2020) Arter97 Kernel, [online]. Available at: https://forum.xda-developers.com/poco-f1/development/arter97-kernel-poco-f1-t3919127

[7] negrroo (2020) LawRun Kernel, [online]. Available at: https://forum.xda-developers.com/poco-f1/development/kernel-lawrun-kernel-v3-t4037729.