Analysis for Performance Optimization of Android Applications

Anirudh Sohil
Bharati Vidyapeeth's College of Engineering, New Delhi
anirudh.sohil@gmail.com

Shikha Rastogi
Bharati Vidyapeeth's College of Engineering, New Delhi
shikha.bvcoe@gmail.com

Rajan Chawla
Bharati Vidyapeeth's College of Engineering, New Delhi
rajanchawla95@gmail.com

***Abstract*— Application developers face many challenges while creating an application for use in next-generation hand-held devices like smartphones, tablets, and smartwatches. As they are very small as compared to actual computers, developers have to keep that in mind that there are limited amount of resources in these handheld devices. So, they have to create an optimized application that not only suffices the need of the user, but as well as satisfy the user with its performance as well. This research paper introduces with the problems encountered by the developer. Also, it aims to provide some reasonable approaches to optimize the applications. We have to take several applications into account as one application could not suffice the need to give us an optimal approach.**

I. INTRODUCTION

While researching information about performance of applications on android platform, we came across a lot of active android Smartphone users who usually try to optimize their application's performance. Users usually try to optimize their application's performance by using some other application, and even deleting some of their log files or even deleting some of their applications which uses up most of the resources of their Smartphone so that they can specifically use a particular application. This research paper aims to establish performance optimization of android application by providing the user with basic knowledge of android application development and how do application's code interact with the operating system.

Performance has been a serious issue with android applications whether it is Android or iOS or Windows. Even in some Smartphone having 2 GB of RAM could not suffice the needs of application. While some of the applications like SD Maid, Greenify and CCleaner have proven to optimize applications a little, but after overtime use, they revert back to the same problem again. With the release of Android 7.1 Nougat, Google announced new features with the new Android like doze, data saver and Vulkan API. But recent polls suggested that not even one percent of android users have migrated to new operating system. This still causes some problem for other 99 percent of the android users. It is because the application made is not optimized at all. Sometimes, the application developers fell short in application development and compromise the performance with the functionality that the application must provide. Performance and functionality do not come hand-to-hand in these applications. An android developer has been tackling this issue for a long time and is facing it from time-to-time.

This research aims at providing an improved solution to the existing problems faced by application developers to improve application's performance no matter what the version of android might be. To achieve this, we have divided this research paper into six sections. In first section, a brief introduction of what an application is and what component it comprises of. This will give a complete idea to the user what does an application comprises of. In second section, a brief account on application framework will be given to provide all necessary frameworks that an application might need. This will give us an overview of what frameworks are used by an application developer to provide the necessary functionality that an application requires as its specification. In third section, we will provide the problem areas that are usually faced by the application developer while developing the android application. This section will cover up all the problems that are hogged up at the time of application development. Due to those facts, the application might not be completely optimized to use on each and every platform that is presently used by millions of users around the world. In fourth section, we will provide our analysis on different basic application which will utilize certain aspects of our Smartphone i.e. hardware and software. Also, we will provide with certain statistics that we have observed on different Smartphone and will compare performance by providing a possible alternative to the problem. And in final section, we will conclude the paper by providing the importance of our project, discussing its future scope and

the importance of expanding this particular project and the work we wish to undertake for implementing the same.

II. APPLICATION AND IT'S COMPONENTS

An android application is simply software that runs the necessary functions provided by the application developer through frameworks, libraries and most importantly, Linux kernel. Android application stands at the topmost layer in the hierarchy of android architecture. This layer utilizes all of functionalities that are provided by the other three layers to it. It is actually what the android as a whole looks to the user.

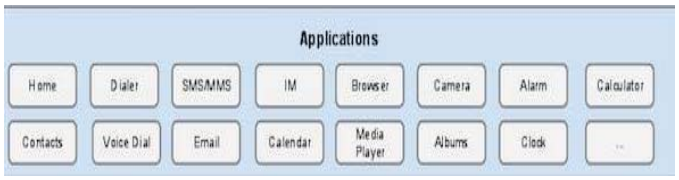

Fig 1. Android Application

Android applications comprises of simple applications that almost each and every Smartphone manufacturer provides with the Smartphone's operating system like launcher, dialer, browser, camera, email and many other. In this layer, the application which developer has made work on. This is the layer which the user actually uses the application he wants to use. This is an important layer as it gives functional and visual overhaul to the user.

An application comprises of several components which perform necessary activities in the application such as activities, views, intents, services, content providers and broadcast receivers. These activities combine together to make an application to perform tasks which it should do [1, 2, 3]. A basic explanation of what an application is shown:

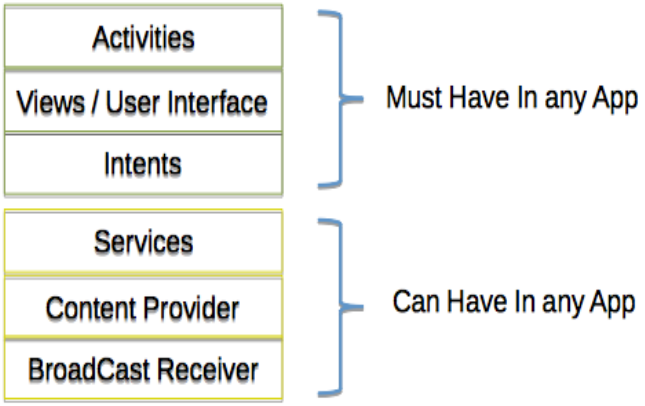

Fig 2. Componets in an android appication [4]

A. Components

1) The activity is the first component of an application which give an app it's functionality within one page. These activities can start a different activity or an intent when they are accesed. They handle how the user interface works.

2) View is just another way how an application is viewed by the user. Views can also be said as user interface of an application. These are used over a combination of activities which give the views it's function .

3) An intent can be described as an abstract description of an operation to be performed. An Intent provides a facility for performing late runtime binding between the code in different applications. Its most significant use is in the launching of activities, where it can be thought of as the glue between activities. It is basically a passive data structure holding an abstract description of an action to be performed.

4) A service is a component that runs in the background to perform long-running operations or to perform work for remote processes. A service does not provide a user interface to the application. It only operates in background.

5) A content provider manages a shared set of app data. You can store the data in the file system, an SQLite database, on the web, or any other persistent storage location your app can access. Through the content provider, other apps can query or even modify the data (if the content provider allows it). Content providers are also useful for reading and writing data that is private to your app and not shared.

6) A broadcast receiver is a component that responds to broadcast announcements. They might opeerate system-wide or it may operate within an application. They arre announced as a broadcast in the platform and are received as an intent in the application.

To provide an app with it's functionality, these componets combine together to form an application which is then used by the user.

III. APPLICATION FRAMEWORK

An application framework is a software library that provides a fundamental structure to support the development of applications for a specific environment. An application framework acts as the skeletal support to build an application. The intention of designing application frameworks is to lessen the general issues faced during the development of applications. This is achieved through the use of code that can be shared across different modules of an application. Application frameworks are used not only in the GUI development, but also in other areas like web-based applications [5].

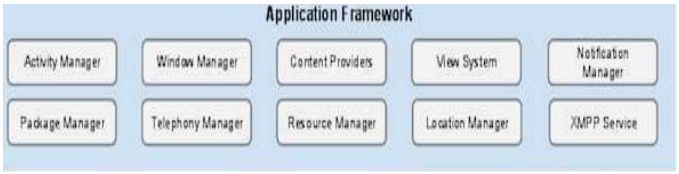

Fig 3. Application Framework

A. Types

1) Activity Manager manages all the aspects that are present within the activity lifecycle and activity stack. All the processes which are neede to perform by an application at that moment of it's working phase are performed by it.

2) Content provider manages a shared set of app data. You can store the data in the file system, an SQLite database, on the web, or any other persistent storage location your app can access. Through the content provider, other apps can query or even modify the data (if the content provider allows it). Content providers are also useful for reading and writing data that is private to your app and not shared.

3) Resource Manager provides access to non-code embedded resources such as strings, color settings and user interface layouts. Through this, many componets of android like unicodes, hex codes for color might be used.

4) Notifications Manager allows applications to display alerts and notifications to the user. It just simply allows the app to display some heads up about some new information provided to it.

5) View System comprises of an extensible set of views used to create application user interfaces. This framework takes care of all visual overhauls that are given to a particular application.

6) Package Manager manages how all the application frameworks and other components will comprise within ana app as a package. It combines all the necessities and bundles them in the form of an package file as *.apk* extension.

Application framework provides these necessary frameworks and many other so that there may be less need to write native code at each and every code in that application [6].

IV. PROBLEM AREAS

While making an android application, we must keep in mind that we are making an application for devices that has less computational power than a computer, and have quite limited power resources to give power to the central processing unit of that device. With this in mind, there are certain other problem areas that are faced by the application developer while developing an android application.

A. Android Versions

Now a days, there are many versions of android out for users on different devices like Jelly Bean, Kitkat, Lollipop, Marshmellow and Nougat the recent out of all. Each operating systems has it's own set of unique functionalities that are provided to it and each has different way of it's operations like Jelly Bean has given full privilidges to the superusers, Kitkat has better RAM management, Lollipop introduced us to Material Design, new in visual aspect of android, Marshmellow gave us with app permissions and doze for battery life and Nougat gave us Vulkan API for better graphics utilization of an application. This is same for other operating systems as well like Windows or iOS [8].

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	1.5%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.4%
4.1.x	Jelly Bean	16	5.6%
4.2.x		17	7.7%
4.3		18	2.3%
4.4	KitKat	19	27.7%
5.0	Lollipop	21	13.1%
5.1		22	21.9%
6.0	Marshmallow	23	18.7%

Fig 4. Android Versions with percentage of it's users on different versions [7,8]

B. Deprecation of some libraries

Due to development of new android versions, it becomes hard to keep backward compatibility for different android versions as it takes up a lot of space. Due to this, some libraries are deprecated which sometimes causes problem for android developer to learn about new directories which have same compatibility as the previous one. This sometimes creates a hassle as it might be possible to re-create an application again due to those particular applications.

C. Limited hardware on some devices

Due to many android devices, it becomes a hassle to make an application which can not only work on different android versions, but can also work on different android versions having their own specific set of hardware. As hardware on a Smartphone is impossible to change, it becomes an important aspect for an application developer not only to work on software, but also on limited hardware.

D. Backward Compatibility

In Android application development, Developers need to think about how they can add new features from the updated Android API's to the older versions and ensuring the applications run as desired on the devices with the older versions. This becomes a hassle for android developers to keep backward compatibility for their applications.

E. Low performance GPU

In android application development, even normal applications demand the GPU usage to render their views. Most of the time, they only use CPU for rendering but in some cases, they might even use GPU. But sometimes in applications like games, simulators or some other graphics oriented application, GPU becomes an important factor. In order for app to run smoothly, it has to run a frame within 36ms (or 60 fps) to run smoothly. But, sometimes an application is not able to render within 36ms which causes application to slow down due to low performance GPU. An application developer has to keep in mind that he needs to run a visual task within a frame of 36ms.

These problem areas accumulate together to create problems for android developers while making their applications for android platform [9, 10].

V. PERFORMANCE ANALYSIS

The android smart phone used for the performance and battery analysis is Redmi 3S Prime. The hardware and software specifications are as follows:

- *Display:* LCD capacitive touch screen, 16M colors, 5 inches (720 x 1280 pixels).
- *OS:* Android OS, v6.0.1.
- *Chipset:* Qualcomm MSM8937 Snapdragon 430.
- *CPU:* Quad Core 1.5 GHz Cortex-A53 and quad-core Cortex-A53.
- *GPU:* Adreno 505.
- *Battery.* Non-removable Li-Po 4100 mAh
- *Internal Memory.* 3GB RAM.
- *Sensors:* Accelerometer, gyro, proximity, compass.

There are two applications we are observing upon the performance of the application and we are testing the applications' performance on standard mobile network on 4G signals. One is a simple map application which uses Google Maps API [11, 14] and other is a simple game based on libGDX [12, 14] and uses OpenGL [13, 14] as base for its graphics rendering. The statistics are measured on above device and is measured by using Android Monitor in Android Studio.

Below are the statistics of the maps application of memory and CPU.

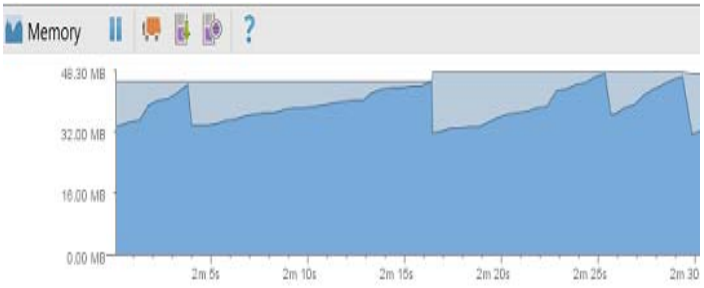

Fig 4. Memory usage of maps application

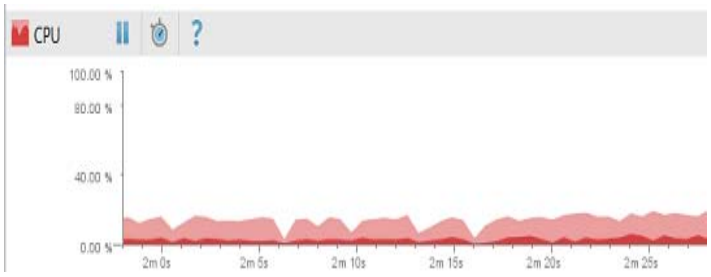

Fig 5. CPU usage of maps application

As we can observe that CPU utilization (in maroon) is less than the CPU allocated (seen in pink) in Fig. 5 to the application. So, in this case the application is using less or adequate CPU. But, as we can see at some points, the app is requiring more memory storage (in dark blue) than the required (in light blue) in Fig. 4 which can be seen by spike at the top indicating more memory heap requirement for the application. So, we can say that this application is CPU efficient but not memory efficient.

A. Reasons for memory spikes

1) Inadequate use of API might have caused fetching of data more than it is required for the application. In this case, we were displaying the map in different views like sattelite, terrain, plain and hybrid. As all are being called at same moment, this might have caused memory spike.

2) Rendering for each and every view of map is taking place in background while one of them is running in foreground. This causes lot of memomy to be used up to create those views.

Now, let us take the statistics of game. In this, we will also take into consideration the GPU usage of the Smartphone. Like other games, this application uses considerable amount of GPU as compared to other application like rendering maps from application. Below are the stats of memory, CPU and GPU of game.

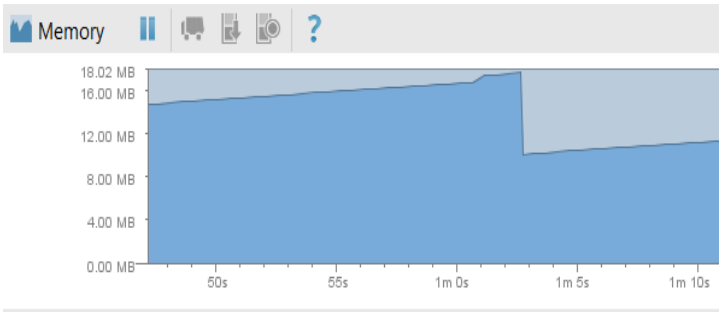

Fig 6. Memory usage of game.

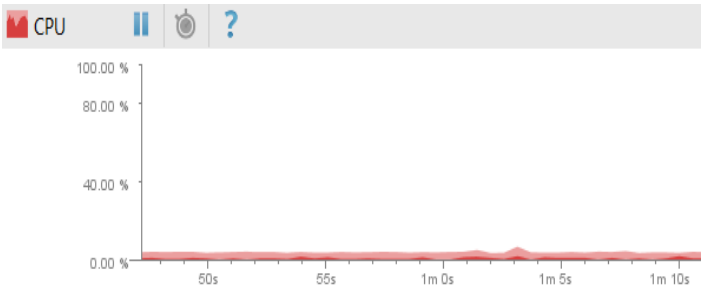

Fig 7. CPU usage of game.

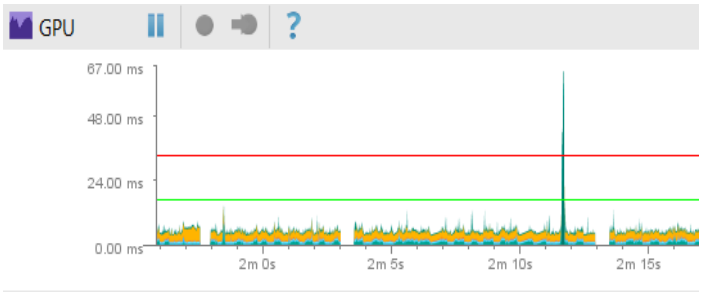

Fig 8. GPU usage of game.

As we can observe, the CPU usage of game is quite less and is not even using 5-10 percent of the CPU. Also, the memory usage is within the allocated range and is not causing it to increase in any way. But, we can see there is a minor spike in GPU usage as it has exceeded the time frame of 36ms causing an app to slow down. If such conditions occur again and again, this might lead to slow running of application.

B. Reason for GPU spikes

1) The software supporting GPU, like OpenGL might not be able to render the game properly. This cause can only be fixed by software vendor.

2) Application developer for that game might not have made it compatible with the previous technology of rendering software. So, a device having previous version of OpenGL might not render that app properly.

VI. CONCLUSION

From examining the usage of both applications, we came to some very conclusive points. From maps application, we came to an inference that loading all data entries from an API at the same time might cause the application to forcefully increase the allocated memory heap. To solve this, we should only necessary aspects from an API that are being called in that view. Not only that, we can limit the loading of data again and again by saving the previous data loaded from an API either as temporary data or as cached data. And from gaming application, we got some results regarding GPU of a smartphone. From that, we examined that at some points, rendering of one frame might not take place within 36ms, which might have caused due to limited hardware or by backward compatibility. From that, we concluded that an application must be made by taking backward compatibility and hardware resources into consideration.

VII. FUTURE WORK

Future work within this area could involve repeating similar measurements on devices running future versions of the Android platform. This would determine whether or not the performance of the applications continues to improve. Right now, we have just improved our application code. But in future, we might make some libraries that you can directly import in your android project or work on some existing open source libraries and plan to improve them. Also, we plan to work on some other applications that can interact more with both API as well as the android operating system. We also plan to work on NDK which might bring some fruitful results on improving the application code.

VIII. GLOSSARY

API: Application Programming Interface

GPU: Graphics Processing Unit

CPU: Central Processing Unit

RAM: Random Access Memory

Li-Po: Lithium Polymer

OpenGL: Open Graphics Library

NDK: Native Development Kit

APK: Android Application Package

IX. REFERENCES

[1] Application Fundamentals , Android Developer, https://developer.android.com/guide/components/fundamentals.html
[2] Bill Phillips and Brian Hardy, “Android Programming: The Big Nerd Ranch Guide” Copyright © 2013 Big Nerd Ranch, Inc., pp. 1-4
[3] Application Architecture, Gartner IT Glossary, http://www.gartner.com/it-glossary/application-architecture-aa/
[4] App Components , Android Developer, https://developer.android.com/guide/components/index.html
[5] Definition - What does Application Framework mean?, https://www.techopedia.com/definition/6005/application-framework.
[6] Zigurd Mednieks, Laird Dornin, G. Blake Meike and Masumi Nakamura, “Programming Android, Second Edition”, pp. 75-86
[7] Platform Versions,developer.android.com/about/dashboards/index.html
[8] The Android Story, https://www.android.com/history/
[9] “Challenges in Android Application Development: A Case Study”, Available at: www.ijcsmc.com/docs/papers/May2015/V4I5201557.pdf
[10] “Sotware Enginnnering Issues For Mobile Application Development”, Available:repository.cmu.edu/cgi/viewcontent.cgi?article=1040&context=silicon_valley
[11] Google Maps API Java,”google-maps-services-java”, https://github.com/googlemaps/google-maps-services-java
[12] libGDX, “libgdx”, github.com/libgdx/libgdx/blob/master/LICENSE
[13] OpenGL ES, developer.android.com/guide/topics/graphics/opengl.html
[14] Apache License, Version 2.0, January 20, 2004, http://www.apache.org/licenses/