International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2017 IJSRCSEIT | Volume 2 | Issue 5 | ISSN : 2456-3307

Building Custom ROM using AOSP and Improving RAM usage in it

Chirag Kanthed[1] , Yagyapal Yadav[2]
Student Department of Computer Science and Engineering[1]
Asst. Prof. Department of Computer Science And Engineering[2]
IES IPS Academy Indore, Madhya Pradesh, India

ABSTRACT

Android which is used most widely all over the world as an operating system after 2013 it not just provide an operating system for mobile devices but it also provides a full software platform and that also includes framework and now it also creates possibilities to use it on much waste range other than the mobile devices. Its first started with Android Alpha version 01.00 and now the latest is Android Oreo version 08.00. Android platform also allows user to edit its actual source code and make changes in it as per the need which can either improve the performance or it can even decrease it. The main intent of this research is to utilize the overall ram usage and to decrease the memory usage by the operating system.

Keywords : Accessing Root , Custom Build ROM , Root , Building ROM , AOSP(Android open source project) , TWRP , Recovery.

I. INTRODUCTION

Android is the largest ever installed base of mobile platform which powers 86.1% mobile phones of entire world and it's based on Linux Kernel. Billions of world devices (approx 2 billion devices) is based on Android. Android is flexible and open to run on different mobile devices with different hardware configuration which increases its popularity among the users.

Its open nature provides freedom to developers to develop applications to fulfill the needs of its rapid increasing user base. Its platform also includes applications , GUI , Middleware and the full operating system. AndroidSDK is a tool which is provided by Google which is used to develop applications. Its openness provides a favorable mindset for users and developers. Various collected data shows Android covers more than 81% around 2 billion mobile devices and for users Android have their own market called play store which contains around 3.2 million applications for users some of them are free to use and some of them are paid. Total 99.8% of total mobile devices is powered by IOS and Android. Android Kernel which process everything is based on Linux. Android provide a powerful framework for development it provide you single model for application which lets you deploy apps to billions of users. It uses different methods than other operating system to handle processes. Android does not shut down every process after its activity ended , it keeps some of its processes alive until more memory is required. Android categories apps into different ways Visible , Content provided , Empty , Foreground , Hidden , Secondary server and its all managed by LMK (Low Memory Killer). It also assign oom_adj value to the processes from range -17 to +15 and kill processes on the basis of oom_adj value of that particular process. Daily approx 1.24 million new devices got activated worldwide in Android platform. Source code of Android is been released by Google. Android devices have extensive variant hardware causes delays in upgrades related to security or new version.

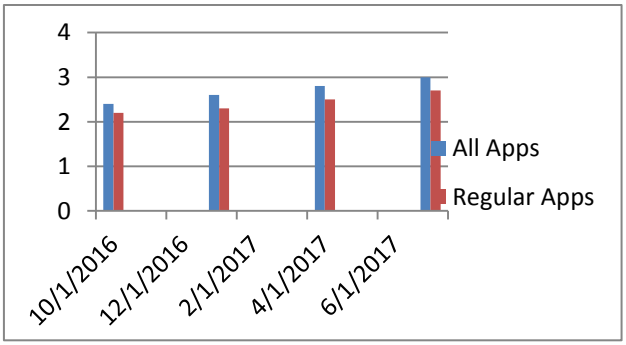

Figure 1. Number of available applications in android market (Millions)

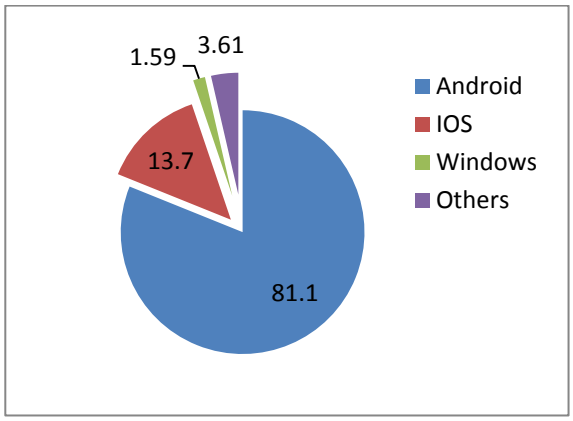

Figure 3. Numbers of Mobile OS Users April 2017 (Millions)

A. Android Availability

For development purpose full source code of around 70gb is freely available to everyone. It simply means you can make your changes in operating system and can use it but it may brick your device but only you are accountable for that. So many users make their changes in the operating system bring new features that may not be available in Android already and Android also apply those features to the official release or in upgrade if that feature is necessary.

B. Android Releases

For each Android version it also has different kernel version and API level. It is an numeric value which uniquely detects the API framework revision which is offered by that particular Android version is known as API level. Coded names of Android From A to O, different kernel version and their different API level given below.

Table 1. Year Wise Android Destribution

Android Version	Code Name	Release date	API level	Kernel Version
1.0	No code name	2008 September	1	2.6.25
1.1	No code name	2009 February	2	2.6.25
1.5	Cupcake	2009 April	3	2.6.27
1.6	Donut	2009 September	4	2.6.29
2.0	Eclair	2009 October	5	2.6.29
2.0.1		2009 December	6	2.6.29
2.1.x		2010 January	7	2.6.29
2.2.x	Froyo	2010 May	8	2.6.32
2.3	Gingerbread	2010 November	9	2.6.35
2.3.1				2.6.35
2.3.2				2.6.35
2.3.3		2011 February	10	2.6.35
2.3.4				2.6.35
3.0	Honeycomb		11	2.6.36
3.1		2011 February	12	2.6.36
3.2		2011 June	13	2.6.36
4.0.x	Ice Cream Sandwich	2011 October	14,15	3.0.1
4.1.x	Jelly Bean	2011 December	16	3.0.1
4.2.x		2012 June	17	3.0.31
4.3		2013 July	18	3.4.0
4.4	Kitkat	2013 October	19	3.10
4.4.x		2013 December	20	3.10
5.0	Lolipop	2014 November	21	3.16.1
5.1		2015 March	22	3.16.1
6.0	Marshmallow	2015 October	23	3.18.10
7.0	Naught	2016 August	24	4.4.1
7.1		2016 December	25	4.4.1
8.0	Oreo	2017 August	26	4.x

C. Android Architecture:

Its architecture consist of six subdivision:

1. ***Stoke Applications*** - Android comes with basic applications so that developers can directly just use that functionality instead of creating such functionality. Such application are Phone , SMS messaging , Email , Calendar , Contacts , Browser etc.

2. ***API Framework*** **-** It's the entire feature-set of Android and it is written JAVA. These API called the core API that will be re used at the time of creation of the application. Each developer have access to those API which is being used in stoke applications. These core API includes

    1. View System
    2. Activity Manager
    3. Notification Manager
    4. Resource Manager
    5. Content Provider

3. ***C/C++ Libraries*** **-** Android core components like HAL and ART are made from code which requires libraries which written in C/C++. By using API framework you can actually use limited functionality of this libraries. To access native libraries Google provided NDK which stands for native development kit. Some C/C++ Libraries are OpenGL | Es , SSL , SQlite etc.
4. ***Android Runtime*** **-** It contains an important component which is called DVM (Dalvik Virtual Machine) designed especially for Android. Dalvik VM allows every application in the system to run their individual process. As in java code compiles to .class or you can say Javabytecode same in Android application create a .dex file which is called Dalvik Executable so that every time application needs to be started it does not need to be compiled again.
5. ***Hardware Abstraction Layer*** **-** The HAL maintains an basic interface which allows API to use the hardware capabilities of device. Whenever an framework API make call to access hardware of the device then the Android systems loads HAL.
6. ***Linux Kernel*** **-** Android platform infrastructure is the Kernel based on Linux. Kernel which actually based on Linux allows Android to gain its main security features and also allows the manufacturers of devices to develop drivers for well known kernel.

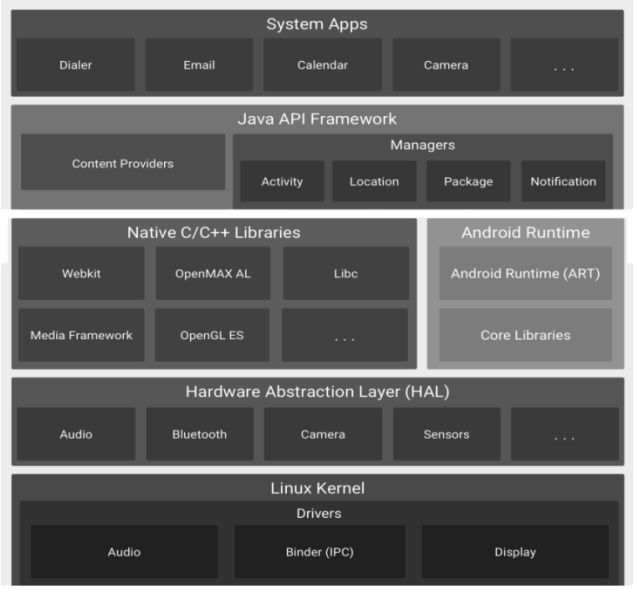

Figure 3. Android Architecture

II. RELEVANT WORK

1. Kumar Vimal & Aditya Trevedi [1] both tried to increase the response time of application. With proper device setting helps in bringing down the response time of the applications and that is beneficial specially for patterns which do not change fast.
2. Deepali Kayande and Urmila Shrawankar [2] used SMSLingo as the SMS application by changing the default Android SMS application which uses UTF - 8 encoding which consumes double memory as compared to the regular normal text. Their experiment results have shown that the average consumption of RAM is least as compared to the Android default. Their experiment can be extended to larges text messages for better and fast result as compared to the Android default one.
3. Abeer Aljarrah and Mohamed Shehab [3] proposed two different approaches one fore system level and other for application level to to enable secure apps for possible floating malicious windows. Application level used to detect the floating window and system level not only detects that malicious floating window but also include an event handler.
4. Saurabh Manjrekar and Ramesh Bhati [4] tries to look into the view behind the Android wheels and keeping spotlight only on custom ROM. We can gain full control over the Android system by rooting it. Rooting is an easy process which gives us the access to so called root. Making changed to

the official ROM result in a custom modified ROM.

5. Jari Kellokoski et al., [5] proposed an power consumption analysis of the best user equipment. Their paper dealt with the analysis of efficient communication in IP data over heterogeneous network. Energy consumption is important because all mobiles devices charged regularly.
6. Tao Ding and Zhenhui Yuan [6] proposed a energy consumption model considering four multimedia based services : voice over IP , file download , web-browsing , video streaming. They studied energy consumption for both TCP and MPTCP and found result that devices using MPTCP consumes more energy as compared to TCP.
7. Ankita Khandelwal & A K Mopatara [7] presented assessment of security for Android with an security architecture overview. They also list various different threats to Android & there countermeasures.
8. Hossain Shahriar et al., [8] performed testing of memory leak of Android applications. First of all they generated some normal memory leak patterns then they generated test cases to find memory leak. Their result shows that their testing approach can find memory leak.
9. Charan K.V et al., [9] takes a look at the various challenges in migration of Android to different embedded devices other then mobile devices. They presented the ways to preparing and building custom ROM in local environment and also preparing Linux-kernel for different platform. They explained how to build own custom ROM from Android source which is open for everyone and then how to transport it to different devices like enterprise desktop, cameras etc. Their success was only possible due to the easy portability of Linux-kernel.

III. PROPOSED METHODOLOGY AND DISCUSSION

Android which is widely used after its been acquired by Google is spreading like a virus it is widely used in mobile devices such as smart phones , tablets , smart watches etc. After its been released it always came up with new features , more storage , more RAM etc. but along with the new functionality the number of problems also increasing one today then two by tomorrow and some of its main problems are

A. RAM – To avoid the extra use of RAM in low memory devices is always required. As per user need user install so many different applications in the system but due to limited RAM some problems may occur , system will perform laggy for that RAM usage must be optimized and that's what we actually did we modified Low Memory Killer Algorithm so that invisible application must be killed so that availability of RAM will increase.

B. Memory – Most of the Android devices comes with limited memory in the system and some of the memory is been used by the operating system as we are building custom ROM the memory requirement for that custom ROM is almost less than half then that of the stoke ROM and rest of memory will not be used so we proposed a memory scheme so that about 1GB memory will be utilized to be used as per user demand.

C. Battery Drain – In Stoke ROM there are lots of API and different application which are not usually required by the user but it is there in the system and that causes battery drain in building custom ROM we will not add those apps which is not required because as we will require those apps we can install it from the Android Market where Thousands of applications are available.

D. Interface – In stoke OS the user interface is quite laggy but in custom ROM we will provide a lag free interface.

IV. CONCLUSION

Building an operating system by Android source is quite challenging and it is really complex. This research determine that all the process we implement to build and customize the ROM will result in optimized RAM and also will result in low memory usage for operating system. Improved RAM usage will result in better application response and good operating system response and more memory will give access to store more where normally will not be able to store any file. We also suggest to users does not make any changes to the operating system if you don't know what its effect going to be. Methods we used will not cause any failure but may have some effect on user experience. So that in our future work we are going to focus on stability of that custom ROM.

V. REFERENCES

[1] K.Vimal & A. Trivedi. "A Memory Management Scheme for Enhancing Performance of Applications on Android " 2015 IEEE Recent Advances in Intelligent Computational Systems (RAICS), pp. 162-166, 2015.

[2] D. Kayande & U. Shrawankar. "Performance Analysis for Improved RAM Utilization for Android Applications" 2015 IEEE Recent Advances in Intelligent Computational Systems (RAICS), pp. 728-733, 2015.

[3] A. Aljarrah & M. Shehab "Maintaining User Interface Integrity on Android" 2016 IEEE 40th Annual Computer Software and Applications Conference, pp. 449-458, 2016.

[4] S. Manjrekar & R. Bhati "CUSTOM ROM – A PROMINENT ASPECTS Of ANDROID" International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) , pp. 1590-1593, 2016.

[5] J. Kellokoski et al., "Power Consumption Analysis of the Always-Best-Connected User Equipment" International Research Journal of Engineering and Technology , 2012

[6] T. Ding & Z. Yuan "Smartphone Energy Consumption Models for Multimedia Services using Multipath TCP" The 11th Annual IEEE CCNC - Multimedia Networking, Services and Applications, pp. 239-244,2014.

[7] A. Khandelwal & A. K. Mopatara "An Insight into the Security Issues and Their Solutions for Android Phones" 2015 2nd International Conference on Computing for Sustainable Global Development, pp. 106-109,2015.

[8] H. Shahriar et al., "Testing of Memory Leak in Android Applications" 2014 IEEE 15th International Symposium on High-Assurance Systems Engineering, pp.176-183,2014.

[9] Charan K.V et al., "Customizing AOSP for Different Embedded Devices" 2014 International Conference on Computing for Sustainable Global Development, pp. 259-264,2014.

[10] Android https:// developer .android .com /index.html

[11] Android http:// Source android. googlesource. com