International Journal of Research in Engineering, Science and Management
Volume-2, Issue-4, April-2019
www.ijresm.com | ISSN (Online): 2581-5792

Optimization of Android Applications: A Survey

Rahul Lotlikar[1], Gajanan Gawde[2]
[1]*Student, Department of Computer Science and Engineering, Goa College of Engineering, Goa, India*
[2]*Professor, Department of Computer Science and Engineering, Goa College of Engineering, Goa, India*

***Abstract*: Android is currently the most popular OS that is used on smartphones. In today's fast world it is necessary that the Applications running on the android platform are optimized. This paper presents various techniques using which many android applications can be optimized.**

***Keywords*: Android, Battery, CPU usage, Network usage, Memory usage, Android Proguard.**

1. Introduction

In today's fast world, the end users have very low patience level. No customer would like to use applications that are slow and/or consume too much battery power. Also the hardware that can be provided in a smartphone is limited because of the size and the weight limitations. There is a need for the software writers to use this hardware as efficiently as possible and write a very optimized code such that the applications run very fast. Basic factors of an application which decide its performance are the CPU usage, the Battery consumption, the Network usage and the Memory usage. If we can optimize CPU, Memory and Network usage, we will also get some optimization on Battery Consumption; i.e. the application will consume a little less power. There are many techniques using which some optimization can be achieved on each of these factors. Many such techniques have been discussed in the following part of this paper. This paper also discusses about Android Pro guard, which is used for making the reverse engineering of the applications very difficult.

2. Literature survey

Each of the factors that affect the application performance and some optimization techniques that can applied on them have been discussed below one by one.

A. Optimizing CPU usage

One of the main characteristics of a good programmer (in any programming language) is: Do not do computations that are not necessary. Same thing also applies in Android Programming (where mostly java is used). One of the frequent mistakes that some programmers tend to make is that they calculate the value of a variable inside a loop, where every time the result will be the same. In such cases it is better to calculate this value outside the loop. Same way, if an expression is repeated multiple times but evaluates to a same value each time, it is better to evaluate it only once, store the value in a variable and use the variable the next time the value of the expression is required.

- Jae Kyu Lee and Jong Yeol Lee [1], Sangchul Lee and Jae Wook Jeon [2], Andreas Ulvesand and Daniel Eriksson [4] have experimentally found out that native code implementations should be preferred over java when it comes to integer calculations, recursions and memory access operations. They also found out that the floating point calculations can be done more efficiently using java. However, Gary Sims [5] experimentally found out that the gap between the performance using Native code and java has drastically reduced since the release of 64-bit Android Marshmallow.
- If inside a loop, data is being fetched from a Global array, the data first can be copied to a local array and then the local array can be used to get the required data inside the loop. This reduces the time spent on lookups [6].
- Use of enhanced for-loops can achieve some performance improvement for devices without a JIT [6].
- Knowing and using the libraries: When using an existing library code, the system has a freedom of replacing calls to library functions with the hand-coded assembler, which in most cases is better than the best code that a JIT produces for an equivalent java.
- If inside a class variables with only constant values are declared, use 'static final' keyword to declare each of the variable since doing this will not require the execution of <clinit> method. The constants directly go into the static field initializers in the dex file.

B. Optimizing memory usage

- Unnecessary objects should not be created. Creation of objects requires some processing and memory. If creation of some objects can be avoided, it should be.
- Do not create temporary objects inside a loop. This will simply consume n times more memory (where n is the number of times the loop runs).
- Use primitive data types wherever possible, e.g., 'Integer' boxed object takes 4 times more memory as compared to 'int' primitive data type.
- Do not keep services running in background unless they are absolutely required. They are memory

expensive.

- Whenever user navigates from one activity to other, release the resources the earlier activity was using. This can be done in onStop and onPause callbacks.

C. Optimizing battery consumption

- If there are any redundant operations which can be cut out, they should be. For example, some downloaded data can be cached instead of repeatedly re-downloading it.
- Some operations which need not be done right away, should be done when the phone has been put for charging. E.g. backing up some data to a cloud.
- Unnecessary services running in the background should be stopped. They consume both the memory and the power.

D. Optimizing network usage

In general, optimizing network usage means reducing the amount of data sent over the network.

- Wherever possible, try to compress the data and then send. This will require some extra processing to extract the compressed data, but comparatively, the extra processing is worth for saving some network traffic and also the battery power required to transfer the extra traffic.
- As mentioned earlier, caching the data which is redundantly downloaded can save a lot of network usage (and battery consumption).

E. Android proguard

Proguard [9] is a Java class file shrinker, optimizer, obfuscator, and preverifier. We can reduce the size of the APK by applying Proguard to our application. The storage space it takes after the installation will also reduce.

- In the shrinking step it detects and removes unused classes, methods, fields, and attributes.
- In the optimization step, the bytecode of the methods is analyzed and optimized.
- The remaining classes, fields and methods are renamed in the obfuscation step using short and meaningless names. Doing this makes the reverse engineering of the application very difficult.
- In the preverification step, the preverification information required for the Java 6 and higher and for the Java Micro edition is added to the classes.

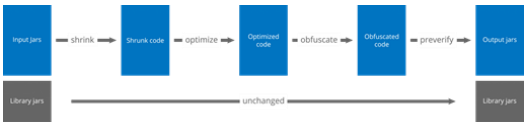

Fig. 1. Proguard Structure [9]

3. Conclusion

By using some good programming techniques, we can optimize our Android Applications. Optimizing for CPU Usage and Network Usage can also reduce battery consumption to some extent. By applying proper memory optimization techniques, we can prevent memory leaks and save a lot of RAM wastage. Using Android Proguard, we can reduce the size of the APK and make the reverse engineering of the application difficult. This increases the security of the application.

References

[1] Jae Kyu Lee and Jong Yeol Lee, “Android programming techniques for improving performance”, in Awareness Science and Technology (iCAST), 2011 3rd International Conference on, pages 386-389, Sept. 2011.

[2] Sangchul Lee and Jae Wook Jeon, “Evaluating performance of Android platform using native C for embedded systems”, in Control Automation and Systems (ICCAS), 2010 International Conference on, pages 1160-1163, Oct. 2010.

[3] Andreaz Lewerentz and Jonathan Lindvall, “Performance and Energy Optimization for the Android Platform,” in Bachelor Thesis in Software Engineering, June 2012.

[4] Andreas Ulvesand and Daniel Eriksson. “Native code on Android: A performance comparison of Java and native C on Android”. Bachelor's thesis at NADA, KTH Royal Institute of Technology, Stockholm. 2011.

[5] Gary Sims, “Java vs C app performance”, https://www.androidauthority.com/java-vs-c-app-performance-689081/, May 2016.

[6] “Performance Tips”, https://developer.android.com/training/articles/perf-tips

[7] “Optimize for Battery Life”, https://developer.android.com/topic/performance/power

[8] “Best Practices for Memory Optimization on Android”, https://hsc.com/Blog/Best-Practices-For-Memory-Optimization-on-Android-1

[9] “Proguard Manual”, https://www.guardsquare.com/en/products/proguard/manual/introduction