International Journal of Scientific Engineering and Science
Volume 8, Issue 3, pp. 11-14, 2024.
ISSN (Online): 2456-7361

Optimize the Performance of Android Mobile Apps Using Android Studio and SDK

Fedorov Stanislav
CEO @ Mobilesource Corp, Miami, United States
Email address: stanislav@mobilesource.com

Abstract— *There are many approaches to creating Android applications. Depending on the chosen strategy, externally identical programs may consume different resources of the mobile device. In other words, their effectiveness may vary. Applications can take up different amounts of RAM, require different amounts of processor time, have different effects on battery power, and download different amounts of data over the network. Ultimately, all these factors affect the perception and performance of your Android application. It is these aspects that distinguish a high-quality application from a less successful one. The main focus should be on performance when designing the application structure. It is also important to take this aspect into account when writing code. A simple example is using Recycler View instead of Scroll View in appropriate cases. The purpose of this article is to review the process of optimizing the performance of Android mobile applications using Android Studio and SDK. The methodological basis was scientific works and research results conducted earlier by other authors.*

Keywords— *Mobile applications, mobile application performance, Android, applications,software, IT.*

I. INTRODUCTION

There are many approaches to creating Android applications. Depending on the chosen strategy, externally identical programs can consume different resources of the mobile device. In other words, their effectiveness may vary.

Applications can take up different amounts of RAM, require different amounts of CPU time, have different effects on battery power, and download different amounts of data over the network. Ultimately, all these factors affect the experience and performance of an Android application. It is these aspects that distinguish a high-quality application from a less successful one.

Performance should be a primary consideration when designing the application structure. It is also important to take this aspect into account when writing code. A simple example is using RecyclerView instead of ScrollView in appropriate cases.

According to statistics from Statista, more than six billion people currently use Android devices, and this figure will grow rapidly in the future [1]. Regardless of the time, place, or method of accessing content, users expect consistent performance within the application. If an app is slow and takes too long to load, most users will likely abandon it. Which in turn makes this topic the most relevant.

The performance of Android apps is important for both creators and users. A productive application ensures smooth operation, minimizes resource consumption and reduces the likelihood of failures, which significantly improves the user experience. In addition, high performance directly impacts user retention and revenue generation. With the increasing variety of Android devices and user expectations, optimizing app performance has become even more critical.

The purpose of this article is to review the process of optimizing the performance of mobile applications on Android using Android Studio and SDK. The methodological basis was scientific works and results of studies conducted earlier by other authors.

II. MATERIALS AND METHODS

Optimizing the performance of Android applications involves identifying key performance aspects and implementing practices to improve the application's performance and requires attention to key performance indicators that are worth identifying and analyzing. These indicators will help you focus your efforts on improving certain aspects, including:

1. Application startup time: Launch speed affects user satisfaction, and exceeding 2-3 seconds can lead to failures.
2. Energy Consumption: Excessive energy consumption drains users' devices, leaving a negative impression.
3. APK Size: The size of the application directly affects the memory space and loading time. Try to reduce the size without sacrificing functionality.
4. Memory Usage: Efficient memory usage is important to avoid slowdowns and crashes, especially on weaker devices.
5. Network Usage: Optimize network requests by reducing network usage, which has a positive effect on application speed and user rates.
6. Frame Render Time: Ensure smooth animations by keeping frame render time within 16ms for comfortable 60fps performance

Next, let's look at existing optimization patterns for improving the performance of Android applications:

1. ViewHolder template for RecyclerView. Implementing the ViewHolder pattern is a performance optimization technique that reduces the number of views created and reused in Android RecyclerView. By using this pattern, you can reduce expensive calls to findViewById(), preventing unnecessary view bloat and speeding up scrolling. In addition, this pattern improves the readability and maintainability of the code.
2. Caching Strategies Effective caching can significantly reduce network requests, improving application performance. Proper use of caching strategies reduces

application dependency on network availability and speeds up data retrieval. Various caching mechanisms, including in-memory, disk, and server-based, can be used in conjunction with appropriate eviction policies such as LRU and FIFO.

3. Lazy loading of images saves bandwidth and improves application performance. Images are loaded only when they are visible to the user, saving resources and providing smoother scrolling for lists with many images. Popular libraries such as Glide and Picasso are widely used to implement this technique.
4. Model-View-ViewModel (MVVM) Pattern Using the MVVM architectural pattern provides a clear separation of user interface logic from business logic and the model, which helps improve maintainability, performance, and code readability. MVVM simplifies data and user interface management, making it easier to handle complex use cases, unit test, and scale applications. Integration of Android architecture components such as LiveData, ViewModel and Room is easy when using the MVVM pattern [2].

III. METHODOLOGY

This article will discuss the following tools that can be used to optimize mobile applications:

1. Android Studio Profiler. Android Studio's profiling tools provide the ability to measure application performance during development. These tools provide data on CPU usage, memory consumption, power consumption, and network activity to help identify and resolve performance bottlenecks.
2. Android Debug Bridge (ADB) Android Debug Bridge is a powerful command line tool that allows communication between a developer's computer and an Android device or emulator. ADB allows you to manage applications, access logs and device information, which is indispensable for identifying and solving problems in an application or system.
3. LeakCanary is an open source memory leak detection library for Android and Java. Using LeakCanary will help identify and fix memory leaks, improving application stability and ensuring a smooth experience. Integration with the library is easy, with minimal effort.
4. Lint is an important static code analysis tool in Android Studio. It monitors the code base, identifying potential vulnerabilities, performance issues, and violations of coding standards. Lint helps improve code quality by identifying problems early in development.
5. Android Studio Network Profiler is useful for measuring your application's network usage and analyzing its interaction with web services. This helps identify ineffective network requests, optimize data flow, and manage bandwidth usage.
6. AppMaster for Android App Development AppMaster is a powerful no-code platform that makes it easy to create high-performance Android apps. Using Kotlin and Jetpack Compose, developers can easily build and deploy scalable applications without the programming headaches. The platform provides an intuitive interface for user interface design and automatically generates source code and Docker containers for server applications, making development more efficient and scalable.

TABLE 1. The tool used in the development of the application and its general characteristics

Tool	Description
Android Studio	An integrated development environment for creating and debugging Android applications.
Android Memory Profiler	A tool for analyzing memory usage in Android applications.
LeakCanary	A library for detecting memory leaks in Android applications.
Room Persistence Library	A library for working with a database in Android applications.
Retrofit	A library for working with network requests in Android applications.
OkHttp	A library for efficiently processing HTTP requests in Android applications.
AsyncTask	A class for performing asynchronous operations in the background in Android applications.
RxJava	A library for programming the event model and managing asynchronous operations in Android applications.
RecyclerView	Widget for effectively displaying lists in Android applications.

The use of these tools contributes to the development of high-performance Android applications [3].

IV. RESULTS AND DISCUSSIONS

In this section, we will look at how to effectively optimize various aspects when creating an application. Optimization of visual elements and interface is achieved through:

1. Reduce the amount of graphic resources: Optimize images using compression and selection of optimal formats.
2. Vector Graphics Applications: Use vector graphics that are scalable and high quality on a variety of devices. These steps reduce system load and allow for more efficient UI rendering.

Optimization of program code and structure is achieved through:

1. Lazy load images: Load resources and data only as needed for faster application initialization.
2. Use multi-threading: Divide tasks into threads to process data in parallel, resulting in increased responsiveness. These methods help your application run smoother and more quickly.

You can optimize memory management by: Data caching: Use caching for frequently accessed data, avoiding repeated requests.

Next, let's look at the capabilities of Android Studio and SDK.

Android Studio's built-in profiling tool allows you to monitor all key app performance metrics: monitor memory usage, CPU load, network activity, and energy consumption.

It is important to note that if obfuscation is applied to an application, the profiling process can be complicated. To monitor CPU usage and identify possible reasons for lag or slow app response, you can use CPU Profiler in Android Studio.

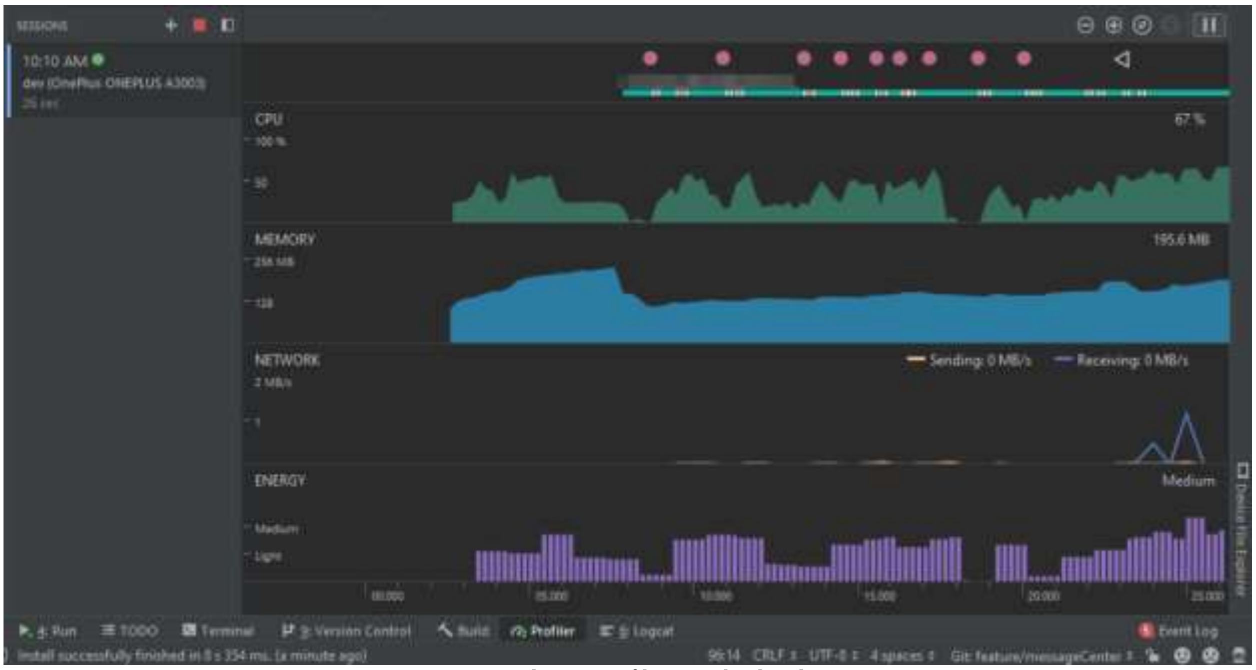
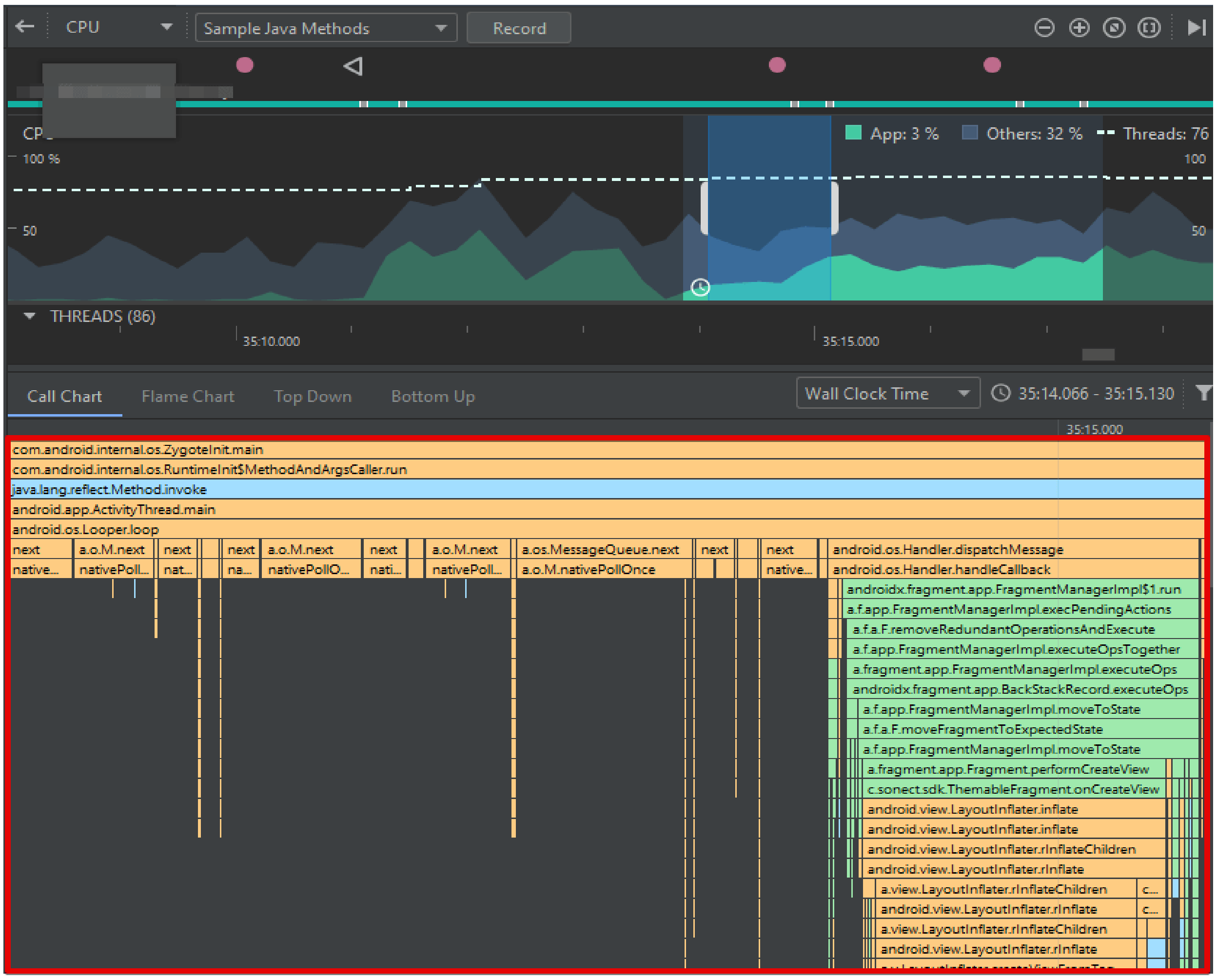
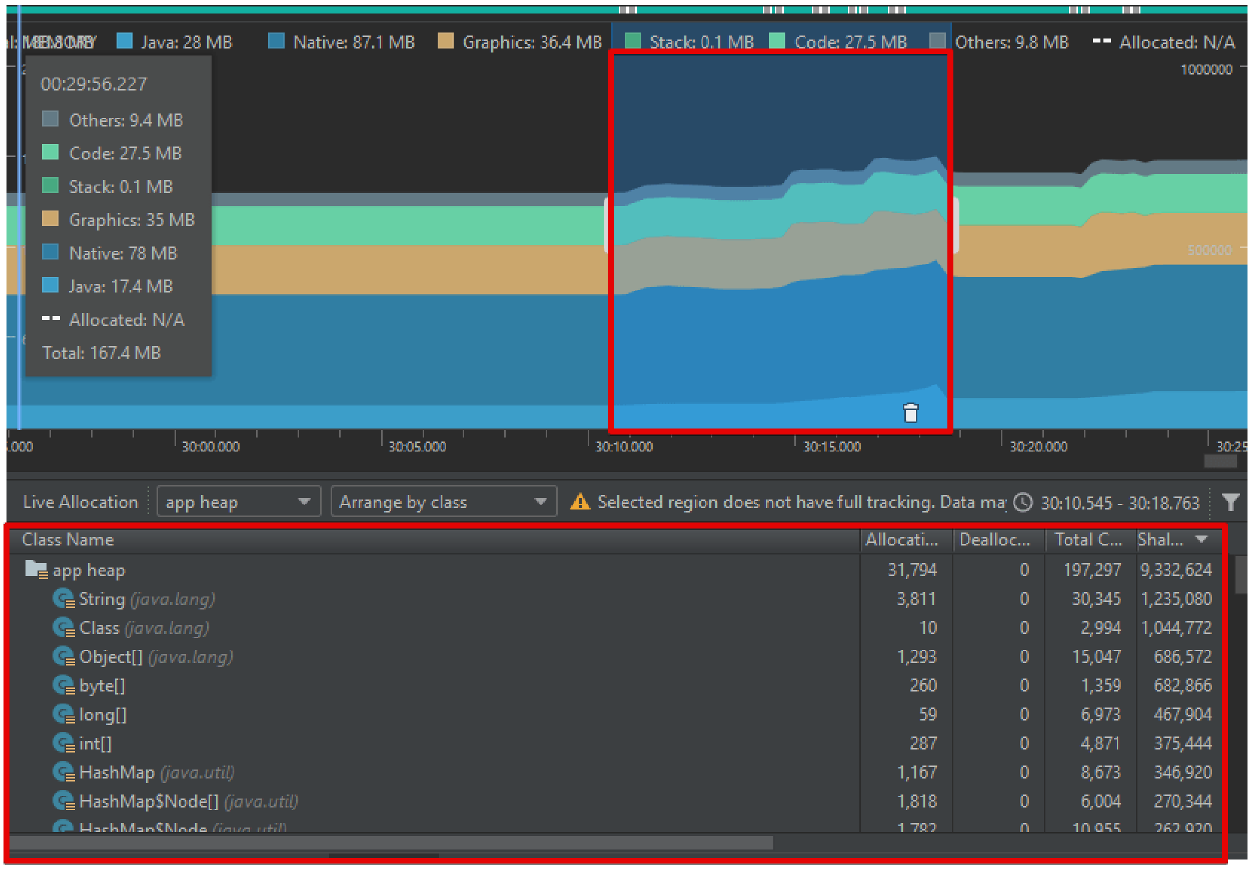


Fig. 1. Application profiler in Android studio



```
com.android.internal.os.ZygoteInit.main
com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run
java.lang.reflect.Method.invoke
android.app.ActivityThread.main
android.os.Looper.loop
next  a.o.M.next  next  next  a.o.M.next  next  a.o.M.next  a.os.MessageQueue.next  next  next  android.os.Handler.dispatchMessage
native...  nativePoll...  nat...  na...  nativePollO...  nati...  nativePoll...  a.o.M.nativePollOnce  native...  android.os.Handler.handleCallback
androidx.fragment.app.FragmentManagerImpl$1.run
a.f.app.FragmentManagerImpl.execPendingActions
a.f.a.F.removeRedundantOperationsAndExecute
a.f.app.FragmentManagerImpl.executeOpsTogether
a.fragment.app.FragmentManagerImpl.executeOps
androidx.fragment.app.BackStackRecord.executeOps
a.f.app.FragmentManagerImpl.moveToState
a.f.a.F.moveFragmentToExpectedState
a.f.app.FragmentManagerImpl.moveToState
a.fragment.app.Fragment.performCreateView
c.sonect.sdk.ThemableFragment.onCreateView
android.view.LayoutInflater.inflate
android.view.LayoutInflater.inflate
android.view.LayoutInflater.rInflateChildren
android.view.LayoutInflater.rInflate
a.view.LayoutInflater.rInflateChildren  c...
android.view.LayoutInflater.rInflate  c...
a.view.LayoutInflater.rInflateChildren
android.view.LayoutInflater.rInflate
```

Fig. 2. An example of executing methods in Android Studio



Selected region does not have full tracking. Data ma 30:10.545 - 30:18.763

Class Name	Allocati...	Dealloc...	Total C...	Shal...
app heap	31,794	0	197,297	9,332,624
String *(java.lang)*	3,811	0	30,345	1,235,080
Class *(java.lang)*	10	0	2,994	1,044,772
Object[] *(java.lang)*	1,293	0	15,047	686,572
byte[]	260	0	1,359	682,866
long[]	59	0	6,973	467,904
int[]	287	0	4,871	375,444
HashMap *(java.util)*	1,167	0	8,673	346,920
HashMap$Node[] *(java.util)*	1,818	0	6,004	270,344
HashMap$Node *(java.util)*	1,782	0	10,955	262,920

Fig. 3. Development tools for creating an application

In Android Studio Profiler, you will notice that sometimes, when actively using the application, the Memory graph is constantly growing. This may indicate a memory leak in the program. Android Studio allows you to see what objects were created while the program was running. Thus, by repeating actions that lead to memory leaks, you can find out what type of objects affects performance [4].

Programmers use a variety of tools to ensure optimal performance. One of the most popular among them is Android Studio, an integrated development environment specifically created for working on Android applications. This powerful tool provides developers with a wide range of useful features, including debugging, code profiling, code completion, and user interface design tools.

Another essential tool is the Android Debug Bridge (ADB), a set of utilities that allow you to interact with Android devices. With it, developers can install and uninstall applications, execute commands from the command line, take screenshots, get logs, and perform many other actions. ADB is becoming an invaluable tool for debugging and testing applications on real devices [5,6].

V. CONCLUSION

Thus, we can say that using Android Studio provides developers with powerful tools for analyzing and optimizing code, as well as improving application performance. The SDK provides access to a variety of tools and resources that help you use device resources more efficiently.

Optimizing applications on Android not only improves their responsiveness and speed, but also reduces energy consumption, which has a positive effect on the energy efficiency of devices. It's important to keep up with new technologies and updates in the world of Android app development to stay up to date with the latest trends and best practices in optimization.

REFERENCES

1. The number of subscriptions to mobile networks for smartphones worldwide from 2016 to 2022 with forecasts for the period from 2023 to 2028 // Statista. [Electronic resource] - Access mode: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide /
2. Nuzhdin D.G. Optimizing the performance of mobile applications: strategies and best practices // International Scientific journal "BULLETIN OF SCIENCE". 2023. No. 11 (68) Volume 3. pp.850-861.
3. Optimizing the performance of Android applications: Tips for creating effective applications. [Electronic resource] - Access mode:https://borisproit.expert/performance-optimization-in-android /.
4. Performance of android applications. [Electronic resource] - Access mode: https://dimlix.com/android-app-performance /.
5. Basic tools for maximizing the performance of Android applications. [Electronic resource] - Access mode: https://forum-media.kz/development-tools/osnovnye-instrumenty-dlya-maksimalnoj-proizvoditelnosti-android/.
6. Performance optimization of mobile applications on iOS and Android. [Electronic resource] - Access mode: https://alfarabifm .kz/advanced-mobile-development/optimizaciya-proizvoditelnost-mobi/ .