Required Parameters' testing for Performance Monitoring and Analysis of Android Applications for various Platforms

Hemadri Nileshbhai Upadhyay
Masters of Technology
In
Electronics and Communication Engineering
Nirma University Institute of Technology

***Abstract*—Because of the rise in Android apps available in app stores and the improved capabilities of smartphones and smart TVs, people are installing more apps on their devices. Additionally, the concurrent use of several high-volume apps is causing the system's performance to degrade. In order to improve the efficacy of the test material, we have identified many key elements in this paper that can be quickly implemented on an Android client and track an application's performance. Some of its main responsibilities include gathering performance data, maintaining performance statistics, and giving feedback.**

Keywords: ADB shell, Performance Testing, live data collection, parameter testing

I. INTRODUCTION

The market is flooded with tens of thousands of applications for mobile and smart TV devices as a result of the introduction and growth of the Android platform. The wireless terminal performance testing, based on the Android platform, continues to be a challenge due to its lengthy nature, high dependency, and challenging optimization. The inconvenient nature of the current wireless testing tool is now cited as the primary justification for lengthening the entire development cycle, harming product upgrades, and making defect management difficult [1–9]. Google's Android operating system, as a leading example of a new generation of operating systems, can garner full attention in addition to the development of mobile handheld devices. This appears to bring more opportunities and problems to the intelligent device industry. Because it is free and open source, Android, which is built on the Linux kernel, quickly gains the interest and support of many manufacturers [2]. At present, the Android system has been dominant in the smartphone system as well as smart TV systems, with many users. Moreover, because the Android application market's audit system is not as complicated as that of iOS and other systems, the Android application market applications have far exceeded those of apple and other systems, resulting in fiercer competition in the Android application market. Therefore, the performance of Android applications is more critical. Today, many companies still get performance data for Android applications by connecting phones/TV to PCs, like iTest professional testing tools, to record apk consumption of CPU, memory, traffic, and power [10]. GT is an app that carries on a tuning platform, an integrated tuning environment that runs directly on the phone with GT, where it can quickly perform the performance testing of the Apps with (CPU, memory, traffic, power, frame rate) [11]. Emmagee is used to perform CPU, memory, network traffic, battery current, and status, rendering real-time processing state in the floating window [12]. This paper mainly focused on the testers' work in the Android terminal and carried out data acquisition and display in Android, as well as required parameter testing to measure the performance and quality of individual android applications for two different platforms such as Android Tv and smart phones.

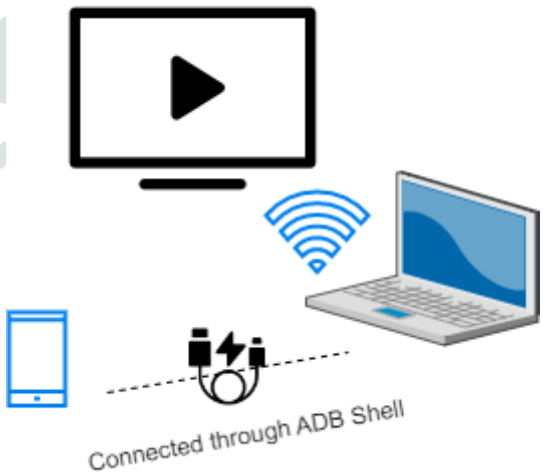
II. SYSTEM REQUIREMENT

A. Mobile as Platform

The number of applications being developed for Android, the mobile operating system with the largest market share, is constantly increasing [1]. System requirements include a web environment with a proper network, like Wi-Fi, mobile data, and 3G or 32G RAM and ROM. The Android operating system has been updated or over graded from version 5.0 or 6.0, and the screen is 5.5 inches in size and has a 1920x1080 pixel resolution. The focus of this paper is on the status and advancement of performance testing based on the market's current apps and users' complaints about such apps.

As shown in above figure, Android smartphone is connected to PC through USB cable. As soon as the tracing process will start, data will be stored in the mobile by itself. Through ADB shell we can pull the data stored in mobile. To analysis and calculations, Monitor connected with PC through Wi-Fi.

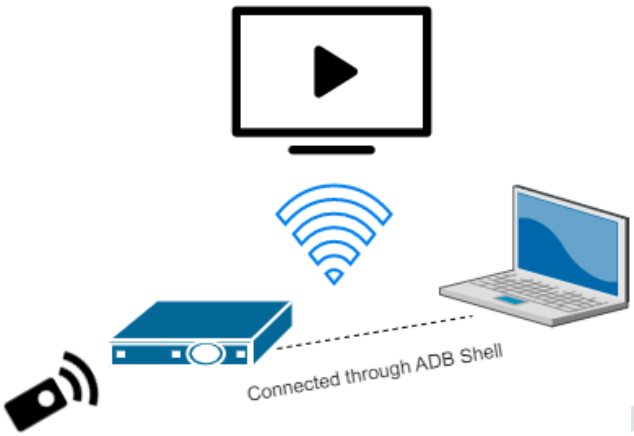
Fig.1 System interface for mobile



B. Smart Tv as a Platform

Android operating system for smart TV is currently growing market. Before integrating applications with Set top boxes, it is important to analyze the performance of individual application. To test the performance of apps for smart TV, we require Set Top box to be connected with PC through ADB. We can directly access the data from STB using ADB commands and can analyze the app performance as well as display the result on monitor.

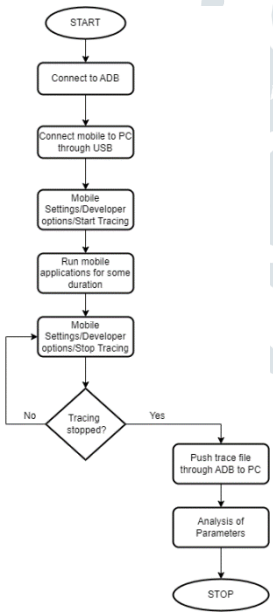
Fig. 2 System interface for Smart TV



III. System Architecture Design

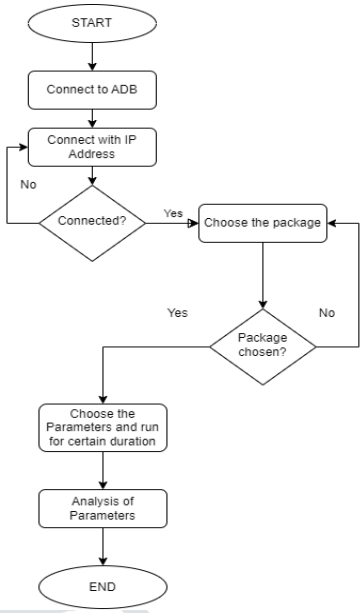
The structure includes data acquisition, data display, and data saving in-memory storage. Performance monitoring module running in the system configuration as shown in Fig 3 and Fig 4.

Fig.3 Flowchart for Mobile App Validation



As mentioned in the flowchart, first of all, we need to connect PC to ADB shell using command-line terminal after that, Android smartphone for which we need to validate the apps is to be connected to PC through USB cable. To start collecting data, we need to open settings/Developer options/start tracing on Android phone. After that, we have to access some applications for certain duration to collect data and that data will be stored in mobile so through ADB command we can pull that data into the PC. For any mobile android application, there are mainly four parameters which we need to monitor to analyze the performance of that app: CPU Footprint, Memory Footprint, Traffic and Power consumption.

Fig.4 Flowchart for Android TV App Validation



According to the flowchart above, the first step in validating a mobile application is to connect the PC to the ADB shell using a command-line terminal. We may connect a PC to a Set Top Box using the ADB shell command, but first we must decide which program we want to test. We must use the application for a set amount of time to gather the data, which will then be saved on the PC and available for analysis. There are additional metrics that must be tracked for Android TV applications, including memory leak, CPU footprint, memory footprint, number of concurrent threads, and memory throughput when switching between programs.

IV. Parameters to be Monitored For Application Validation For Smartphone and Android TV

A. Parameters for Mobile Applications:

Parameters	Explanation
CPU Footprint	CPU footprint or usage is the percentage of the amount of time a CPU spends processing non-idle tasks. It has the following characteristics: Constantly changing: A switch's CPU usage keeps changing with system operations and changes of the environment. Non-real-time: CPU usage data reflects CPU usage within a statistical period. A heavy load with only a few running programs may indicate insufficient CPU power support, or running programs hidden by the system monitor - a high indicator of viruses and/or malware.
Memory Footprint	The term "memory footprint" describes how much of the system's main memory a running software utilizes or refers to. The memory footprint of a software program in computing represents the amount of

	runtime memory needed to run the program. This includes all of the active memory areas, such as the code segment containing (mostly) program instructions (and sporadically constants), the data segment (both initialized and uninitialized),[1] heap memory, call stack, as well as memory needed to hold any additional data structures, such as symbol tables, debugging data structures, open files, shared libraries mapped to the current process, etc., that the program may ever need while executing and will be loaded at least once during execution.
Network Traffic	The network hardware spends a large amount of time idle while an app is using the network resources efficiently. On portable devices, turning on the radio to send or receive data and keeping it on for extended periods of time incur considerable costs. If your app is effectively utilizing the network, you should notice that its network communications are closely spaced, clustered together, and include intervals during which the program is not requesting a connection.
Power Consumption	Sustainability of Android apps depend on the power consumption of the apps. Developers lack the knowledge to enhance the energy efficiency of applications as they are not aware of their energy demands during development.

B. Parameters for Android TV Applications:

Parameters	**Explanation**
CPU Footprint	CPU footprint, often known as CPU use, is the portion of time a CPU is processing non-idle operations. It features the following things: continually evolving: The CPU consumption of a switch fluctuates as a result of system activity and environmental variables. Non-real-time: Data on CPU utilization shows CPU activity over a predetermined time frame. A major sign of viruses and/or malware is a heavy load with only a few operating processes or running programs that are hidden from view by the system monitor.
Memory Footprint	The term "memory footprint" describes how much of the system's main memory a running software utilizes or refers to. The memory footprint of a software program in computing represents the amount of runtime memory needed to run the program. This includes all of the active memory areas, such as the code segment containing (mostly) program instructions (and sporadically constants), the data segment (both initialized and uninitialized), heap memory, call stack, as well as memory needed to hold any additional data structures, such as symbol tables, debugging data structures, open files, shared libraries mapped to the current process, etc., that the program may ever need while executing and will be loaded at least once during execution.
Memory Leak	When an application allocates memory for an object but forgets to release the memory when the object is no longer needed, a memory leak occurs. Leaked memory builds up over time, which causes sluggish app performance and even crashes.
Number of Threads	The preferred technique for maximizing performance from multi-core CPUs is threading. It could appear that if some threading is beneficial, then more must be even more beneficial. In reality, a program might become sluggish from having too many threads. Too many threads might have two negative effects. First, when a fixed quantity of work is divided among too many threads, each thread receives so little work that the overhead associated with initiating and stopping threads overwhelms the productive work. Second, running an excessive number of threads results in overhead due to the way they compete for limited hardware resources.
Memory Leak during App Switching	Memory leaks occur when an object that is supposed to be garbage collected has something holding a reference to it. As more and more instances of that object are created, older instances are still being retained in the application's memory. A memory leak is the gradual deterioration of system performance that occurs over time as the result of the fragmentation of a computer's RAM due to poorly designed or programmed applications

	that fail to free up memory segments when they are no longer needed.

V. Tools Which are Currently Available to Check Application Performance

A. Android Monitor

This tool, which was produced by Android Studio, is by default located in the bottom left corner. In the Android Studio, there are two tabs: Logcat and Monitors. These tabs are simple to flip between. Network, GPU, CPU, and memory graphs are among the four different graphs included in the monitor area. Additionally, each component performs a unique purpose, the bulk of which are self-explanatory.

B. *Hierarchy Viewer*

One of Android Studio's most useful tools is this one. You can retrieve a clear and organized summary of your view setups using Hierarchy Viewer. For improved app speed, you may also optimize your app's user interface (UI) by experimenting with various design layouts and unique perspectives. This tool assists you in identifying performance issues brought on by the hierarchy of your views. The hierarchy can then be streamlined and overdraw can be decreased. The Android Developer website provides an explanation of the various panes and further information about this tool.

C. *Profile GPU Rendering*

One of the best Android app optimization tools for performance debugging is Profile GPU rendering, which can be accessed through Developer settings. "On screen as bars" should be selected after selecting this option. When you've finished, colored bars will appear on your screen to show distinct application features and their respective benefits. Through this tool, you may also determine if any frames have been missing.

D. *GPU Overdraw*

After fostering developer mode, you can enable this useful tool from the developer option. When this tool is activated, you can choose Debug GPU overdraw, “Show Overdraw Spots”. Next, the screen will fetch some strange colors to highlight the number of times a specific section was overdrawn.

VI. Conclusion:

Failing to ensure optimal performance of apps can cost organizations huge financial losses, compromised brand reputation, low conversion

The key reasons why organizations should conduct web app performance testing are—

- Determining performance bottlenecks: Performance bottlenecks can cause software crashes or bad user experiences. Examples include slow reaction times, poor scalability, system downtime, battery drain, etc. Before the product launches on the market, testers can easily identify performance problems through meticulous performance testing and investigate money-saving solutions.

- Ensuring stability across multiple platforms: Stability across many platforms is ensured by the fact that users access web apps using a variety of browsers, gadgets, and operating systems. The effects of these various platforms vary on factors like load time, response time, and other important performance metrics. As a result, testing teams employ efficient performance testing techniques to guarantee that the apps operate flawlessly across all platforms and provide an excellent user experience.

- Verifying compliances: For all project stakeholders, the data gleaned from the performance testing is essential. These make it possible for teams to design a configuration that offers high-quality performance without compromising business goals and assist in achieving compliance with SLAs, contracts, and current legislation.

- Discovering ideal configuration for applications: Performance testing of web apps assists in predicting how different system resources are required to maintain the applications. Post the test, the project teams can determine the configuration that delivers high-quality performance with no compromises to business objectives.

- Reducing revenue losses: Project teams can lower the likelihood of downtime and lower the costs related to it by recognizing the risks and vulnerabilities of system shutdowns and optimizing the web app.

Acknowledgment

I would like to express my gratitude and sincere thanks to **Prof (Dr.) N.P.Gajjar**, PG Coordinator of M.Tech Embedded Systems for guidelines during the review process.

I take this opportunity to express my gratitude and deep regards to **Assi. Prof. Mr. Jayesh Patel**, guide of my internship project for his exemplary guidance, monitoring and constant encouragement.

I would also like to thank **Mr. Yakasiri Jayachandra**, external guide of my internship project from **Vantiva**, for guidance, monitoring and encouragement regarding the project.

References

[1] Xue, D. 2015. Design and Implementation mobile application performance monitoring system based on Android. Xidian University of Electronic Science and Technology.

[2] Chenhui Xie, Jian Zhou, ShanShan Li ,Lu Ying Jia , The Design and Implementation of Mobile Monitring System of Transmitting Station based on Android Platform,[D] 2012 International Conference on Mechnical and Electronics Engineering, Beijing,2012.

[3] Qi Luo, A. N. 2017. FOREPOST:findind performance problems automatically with feedback directed learning software testing. Empir Software Eng (2017)22.6.

[4] Wen, Design automation software for Android test system performance and implementation Of [D], 2016.

[5] Wei, Explore the development of automated software testing tools under Android platform [J] , (2015) 30 (2): 155.

[6] Lukun, Research and Implementation Android smart phone performance automated test system [D] Beijing, 2017.

[7] Yun, performance of key technology of software test platform research and application of [D] Beijing, 2017.

[8] Xingnong, Android-based APP automated test platform design and implementation of [D] Dalian , 2016.

[9] Warren, I. and Meads, A. (2017) Towards a Technology Agnostic Approach to Developing Mobile Applications and Services. Journal of Soft-ware Engineering and Applications, 10, 500-528.

[10] Ashwaq A.Alotaibi, R. J. "Novel Framework for Automation Testing of Mobile Application using Appium". International Journal of Modren Education annd Computer Science (IJMECS),Vol.9,No.2,pp.34-40, 34-40.

[11] iTest. 2018. Testing Tool https://soft.shouji.com.cn/down/29068.html

[12] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of GT.(n.d.).TestingToolRetrievedfromhttps://blog.csdn.net/harryzzz/article/details/81381920