A Study on the Performance of Android Platform

Hyeon-Ju Yoon
Department of Computer Engineering
Kumoh National Institute of Technology
Gumi, Republic of Korea
juyoon@kumoh.ac.kr

***Abstract*— As the Android platform is widely used for embedded systems including smart mobile devices, the needs for systematic performance analysis have significantly increased. System performance is usually measured by benchmarks and profiler software. We studied on the performance of Android platform using a benchmark application and public profile software. For more detail and integrated performance analysis, we proposed a profiling architecture of Android platform.**

Keywords-Android; performance; benchmark; profile;

I. INTRODUCTION

Smartphones and tablet PCs are making big change in our life these days. The most popular operating systems for smart devices are Apple's iOS and Google's Android. Because Android is open source software, and offers developers free platform to make their own applications, lots of hardware vendors adopt Android and market share is also increasing.

Even if the platform is common and has the same software capability, the actual performance varies with hardware and other software components. So every hardware and software developers make great efforts to achieve higher optimized performance. To release a smart phone product with some preferable house software, developers should estimate the performance in detail so as to correct or enhance the weak points.

In this paper, we looked into two kinds of software tools for measuring system performance, benchmark and profiling software. Benchmarks are useful for evaluating and estimating the relative level of each device and overall system, so can help us choose hardware or adjust system variables to achieve higher performance. Profiling software traces the program activities and gathers information about function calls, memory usage, process, and communication. After we introduce representative benchmark and profile software respectively, we propose a modification for standard tool and show a simple performance analysis result. These will be a basis for future work to develop integrated and comprehensive performance analysis software.

II. ANDROID PLATFORM

Android is a software stack for mobile devices that includes an operating system, middleware and key applications [1]. It is developed and maintained as an open source project led by OHA (Open Handset Alliance) [2], which aims at building a better phone for consumers. The Android architecture is shown in Fig. 1.

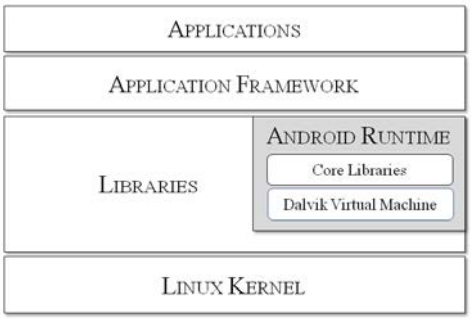


Figure 1. Android Architecture

The base system of Android architecture is Linux kernel 2.6. It supports security, memory management, process management, network stack, and device driver model.

A set of C/C++ libraries is used by various components of the Android system. They consist of standard C system library(libc), media libraries including MPEG4, H.264, MP3, JPG, and PNG, surface manager for display subsystem, LibWebCore as a web browser engine, 2D graphics engine SGL, 3D graphics libraries, FreeType for font rendering and SQLite, a lightweight relational database engine.

Android runtime includes a set of core libraries that provides functionality of Java programming language. Dalvik virtual machine supports a runtime environment for Android Java applications. Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Conventional Java virtual machine is a stack-based machine, but Dalvik is register-based and executes files in Dalvik Executable(.dex) format.

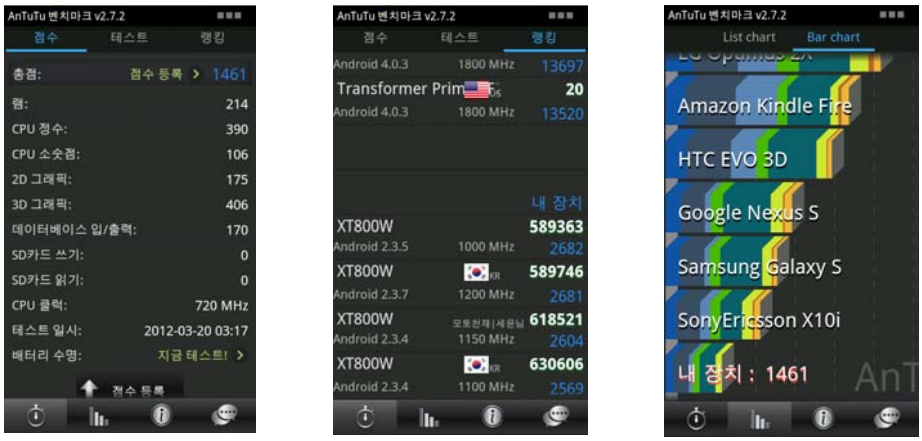
Application developers usually access to the application framework layer through lots of APIs while they develop programs with Java and XML. Application framework is an open software development platform that includes view management, content providers, resource manager, notification manager, and activity manager. Many reusable components are released in the framework, and the developer can replace the components or publish its own capabilities.

III. OFF-THE-SHELF PERFORMANCE ANALYSIS TOOLS

There are several kinds of tools for evaluating and analyzing the performance of systems or applications. We studied and tested some benchmark tools and performance measurement software for application developers. They can be acquired from the open marketplace or open websites for free.

A. Benchmark

Benchmark tool is a programming application that evaluates or gauges the relative performance of a system. It runs a special program on the target device and system, gathers the performance data, and shows them as a quantitative value.



(a) score (b) ranking (c) comparison of devices

Figure 2. AnTuTu Benchmark results

Fig.2 shows the results of a well-known benchmark application, AnTuTu-Benchmark [3]. It can run a full test of a key project, through the “Memory Performance”, “CPU Integer Performance”, “CPU Floating point Performance”, “2D 3D Graphics Performance”, “SD card reading/writing speed”, and “Database IO Performance” testing. The final score represents a relative value of the tested system and can be compared with other devices’ results. Fig.2(b) shows the ranking among the same kind of devices, and Fig.2(c) depicts the relative performance among different kinds of devices. We can see that even the same kind of devices may show the very different performance according to the hardware tuning status and system software version.

Quadrant standard edition [4] and SmartBench [5] are another well-known benchmark application for Android devices, which can measure overall performance like AnTuTu. The other kinds of benchmark applications such as CF-bench [6], GLBenchmark [7], Linpack [8], BenchmarkPI [9], are used for a specific area of system, for example, CPU or graphics subsystem.

Benchmark applications are good tools for evaluating and estimating the relative level of each device and overall system, so can help us choose hardware or adjust system variables to achieve higher performance. However, it is difficult to indicate which part affects the performance or which part we should manipulate for the better performance. The detailed software performance analysis is also not available from the benchmarks.

B. Android SDK Tools

In the Android SDK(Software Development Kit), several software tools are included for assisting developers with debugging, monitoring, and profiling. Some of them can be used for performance analysis. Most useful and convenient tools are DDMS(Dalvik Debug Monitor Server) and Traceview because they provide the graphical view.

DDMS a debugging tool with graphical interface, which provides port-forwarding services, screen capture on the device, thread and heap information on the device, logcat, process, and radio state information, incoming call and SMS spoofing, location data spoofing, and more. While developing with Eclipse, we can open the DDMS perspective as shown in Fig.3.

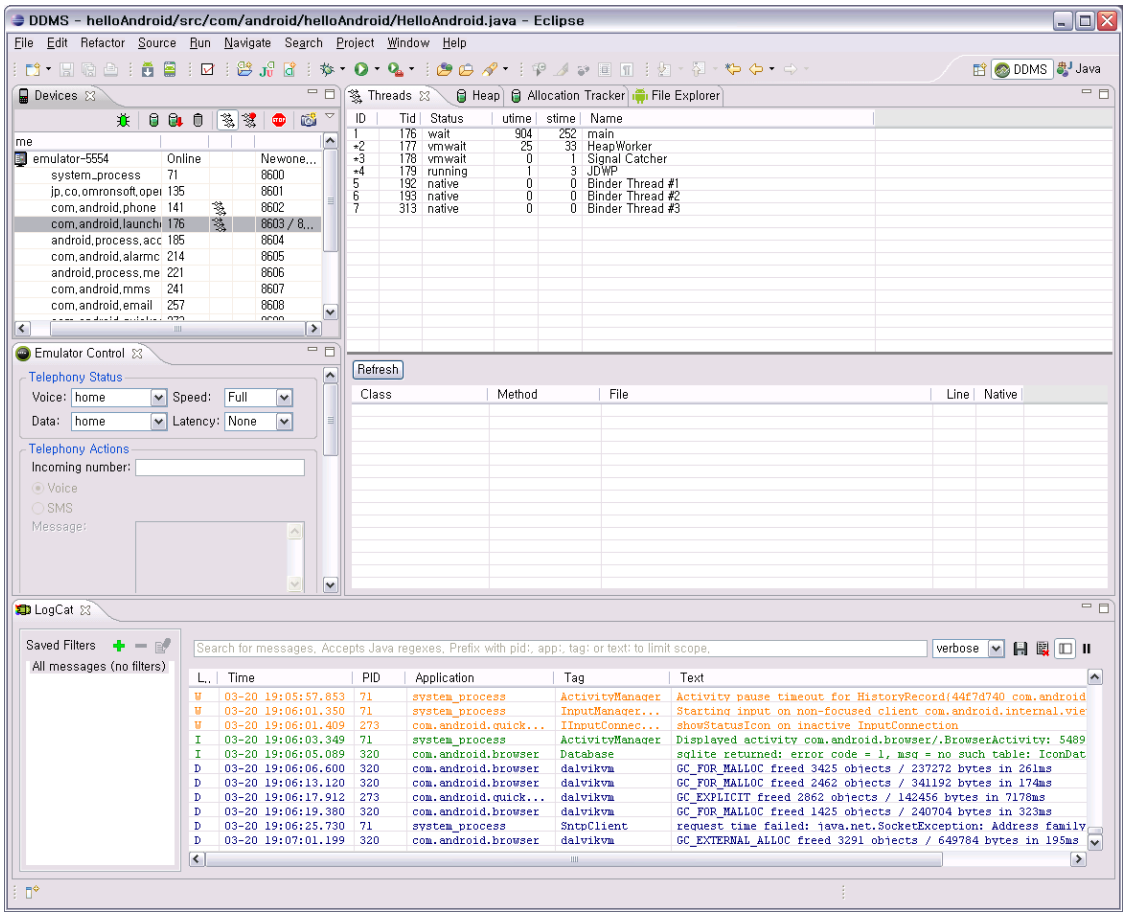


Figure 3. Screenshot of DDMS

Of the debugging tools, method profiling tool is useful for tracing the flow of operations and duration time spent executing the methods. It gathers method calls and estimates the execution time while we interact with applications. Method profiling is invoked and ended with menu “Start Method Profiling” and “Stop Method Profiling” of DDMS or program code startMethodTracing() and stopMethodTracing() of Debug class. The results are recorded in a log file and sent to *Traceview* tool which displays the logs graphically as shown in Fig. 4 and 5.

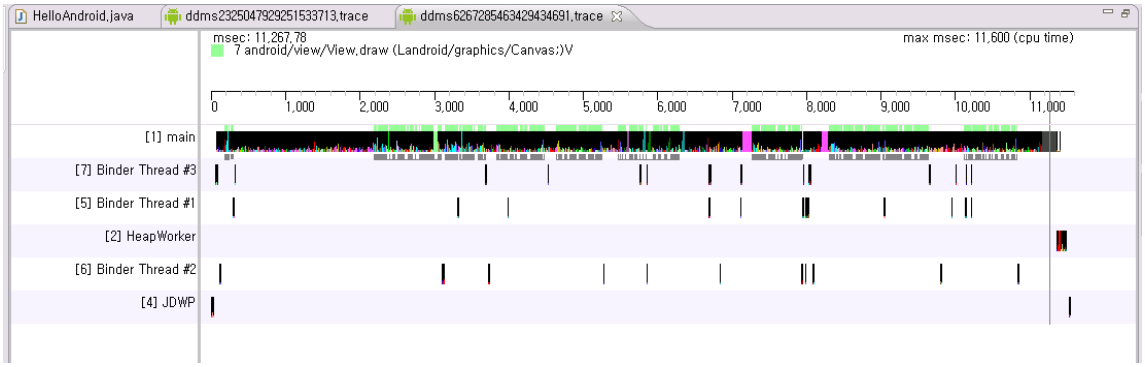


Figure 4. Traceview timeline panel

In timeline panel, each row represents threads with time increasing to the right. Each method is shown in different colors which are used in round-robin pattern. If we select a method, we can see its log record in the profile panel (Fig. 5). Profile panel shows exclusive execution time, inclusive time (with called functions), and portion of total execution time. Total number of calls and number of recursive calls are written in the last column.

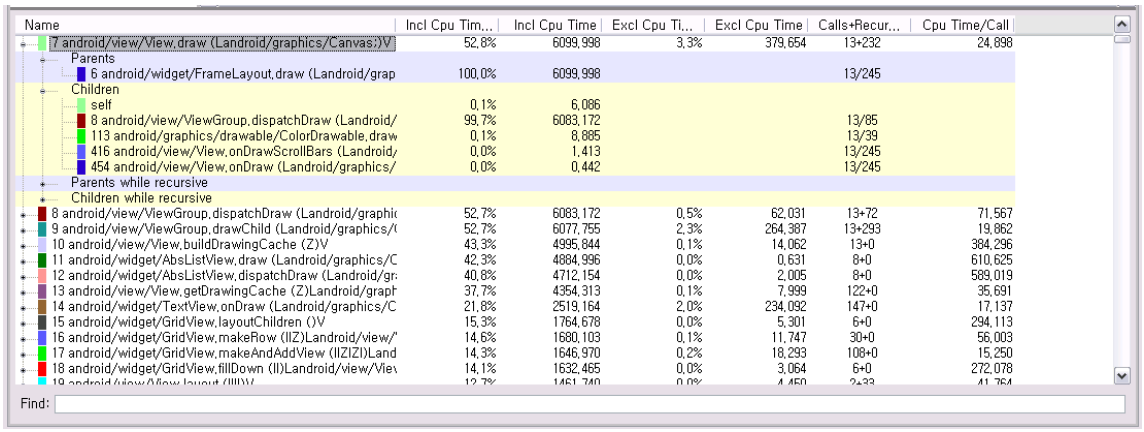


Figure 5. Traceview profile panel

IV. Profiling Tool and Analysis Results

For smartphones, absolute speed is important issue but the responsiveness is more critical for user satisfaction. In addition to game applications which require complicate 3D graphic operations, even in simple web browsing or address book, memo note apps, users may feel slow response to their touch input. We modified DDMS and Traceview to make analysis procedure a little faster, and analyzed view system performance of Android framework for Android 2.2 (Froyo).

A. Modification of DDMS

Although the Traceview offers nice graphical user interface, it is sometimes intolerably slow because it is written in Java and runs as an eclipse plugin. To achieve better profiling speed, we decomposed the Traceview into log data processing part and display part and newly implemented *Pretrace* program which processes the log data. Call records and analysis on the start and end time are created and analyzed by Pretrace, and Traceview displays the results in timeline and profile panel. Fig. 6 shows the structural diagram of our modification.

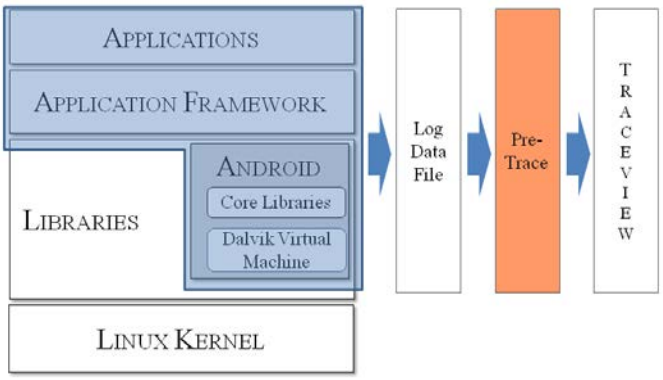


Figure 6. Modified method tracing

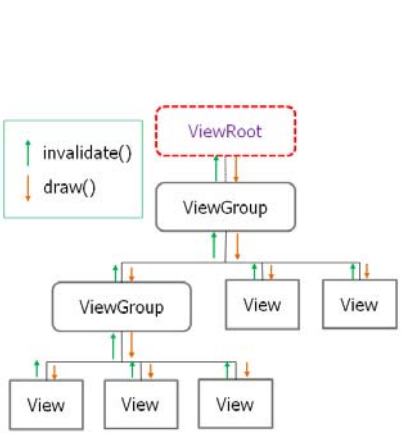
Another problem with DDMS is that it focuses on the application and application framework based on the internal behavior of Dalvik virtual machine, so the native library, Linux kernel and overall integration effect on performance cannot be observed in detail. But this requires introduction of other tools or new implementation of complicated profiling software, we established it as a future research topic.

B. View System Performance

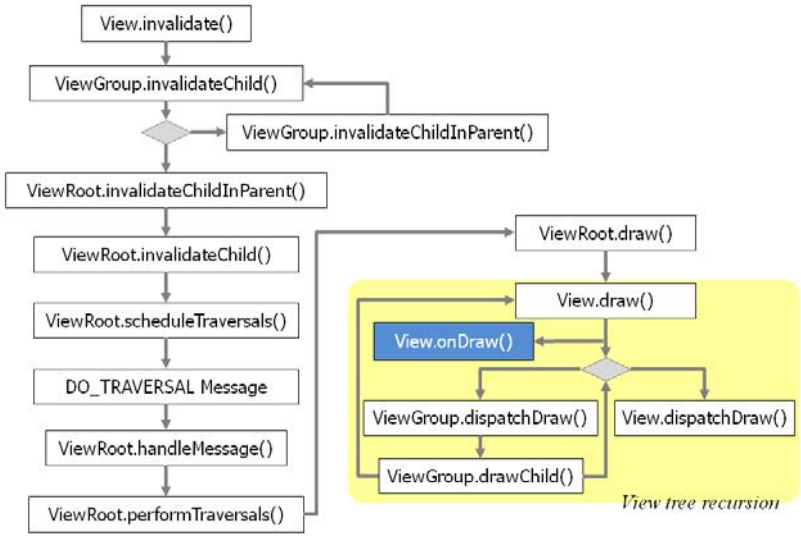
Poor responsiveness may be caused by several reasons. Froyo, our experimental platform, was known as that suffered from event delivery mechanism and poor system dynamics. (It is said that Gingerbread improved the system dynamics significantly [10].) Garbage collection mechanism is also not so good because of the synchronization problem among threads and meaningless waiting time. We focused on the view system to analyze the responsiveness performance.

With the help of the method call records and *dmtracedump* tool, we can depict the view system execution structure as in Fig. 7. According to the view hierarchy, existing views are invalidated, and new canvas is constructed from layout objects. Actual drawing and displaying occur when the *View.onDraw()* method is called. Objects are in tree structure, and the view system traverses the tree and draw child objects recursively.

The problem is the recursion is executed in single thread, so total rendering time can be very long, resulted in skipping frames or stopping animation. When we observe the portion of each method in profile panel of Traceview, we see that many applications spend largest time in view system and rendering. Moreover, many of recent smart devices use multi-core CPU. Because recursion in single thread cannot utilize of high performance of multi-core high-end CPUs, other view system mechanism is needed for better performance.



(a) View hierarchy



(b) View system execution flow

Figure 7. View System

V. CONCLUSION AND FUTURE WORKS

We reviewed some software tools for analyzing the performance of Android platform. Benchmark applications are useful for estimating relative device performance and can be used for tuning and adjusting the performance variables. For application and framework performance, we can utilize the Android SDK tools such as DDMS and Traceview.

With modified debugging and method trace tool, we analyzed the performance of Android view system. Recursive view tree traversal may slow down the rendering process, and sometimes cause cut between smooth animations on display.

We have only worked for the Android 2.2 Froyo version. Recent versions such as Gingerbread(Android 2.3) and Ice Cream Sandwich(Android 4.0) are reported as they significantly improved the system dynamics and some time-consuming components, so resulted in better performance. We will test more programs on newer platforms, analyze the performance degrading factors. We expect the result can help the performance improvement. Another future topic is integrating the Linux kernel profiler, such as Ftrace [11], with Android profiler programs because they are both excellent system but their harmony in operation needs to be examined more for performance issue.

ACKNOWLEDGMENT

This paper was supported by Research Funds of Kumoh National Institute of Technology.

REFERENCES

[1] http://developer.android.com
[2] http://openhandsetalliance.com
[3] http://www.antutulabs.com/AnTuTu-Benchmark
[4] http://www.aurorasoftworks.com
[5] http://smartphonebenchmarks.com
[6] http://www.chainfire.eu
[7] http://www.glbenchmark.com
[8] http://www.greenecomputing.com/apps/linpack
[9] http://AndroidBenchmark.com
[10] “Android 2.3 platform highlights,” http://developer.android.com/sdk/android-2.3-highlights.html..
[11] http://elinux.org/Ftrace

AUTHORS PROFILE

Hyeon-Ju Yoon is working as an assistant professor of Department of Computer Engineering at Kumoh National Institute of Technology, Korea since 2005. She earned the Ph.D. degree in computer engineering from KAIST(Korea Advanced Institute of Science and Technology) in 1997. Her research areas of interest are computer systems including operating system, distributed system, and embedded systems.