CPU and Memory Monitor

Behnam Omidi

ECE 516 Spring 2021

I. Introduction

Monitoring the CPU and Memory usage in mobile devices assists the user to identify CPU and Memory intensive apps consuming more power and deactivate them in critical situations such as when the battery is low or the mobile needs more speed to deal with an app. AnotherMonitoring is a mobile system performance monitoring utility allowing the user to monitor the average CPU and memory usage every 0.5, 1, 2, or 4 seconds. In this project, AnotherMonitor has been modified in a way that the user can change the resolution of updating the values every 0.2, 0.5, 1, 2, 4, 8 seconds and track the utilization of every core inside the mobile. The rest of the document is organized as follows: section II gives a brief description of the tools leveraging in this project. section III introduces how we can modify char style in the application. Section IV shows how resolution changing is achieved in the program and section V is dedicated to changes that occurred in the app for reporting every core usage.

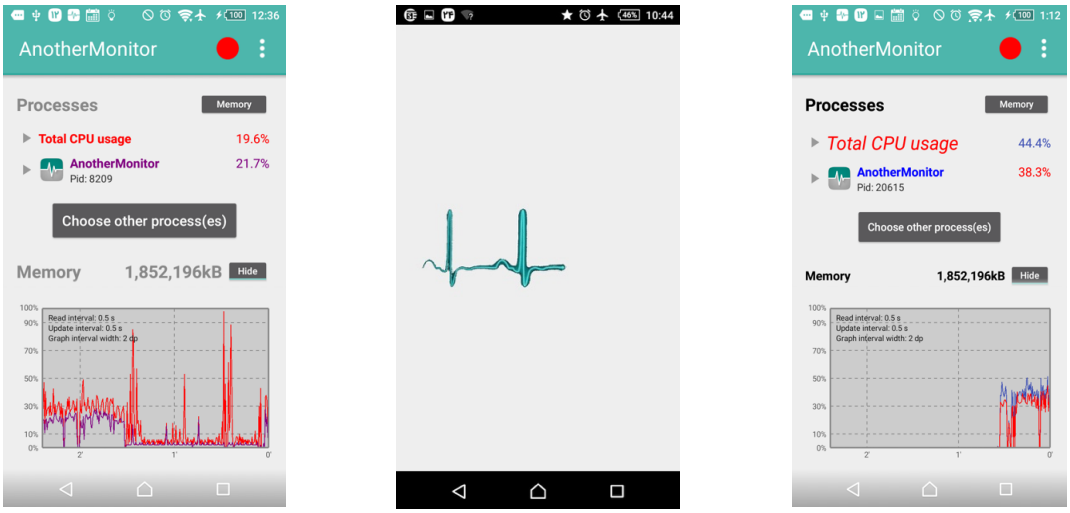
You can find phases of project here: https://github.com/beomidi/ECE516

II. Tools:

Android studio and SDK version 30 are exploited for building the original AnotherMonitor app. A Sony Experia ZR with android 5.1.1 (lollipop) is used as a real platform to execute and test the modifications in the app.

III. Char Style:

We can modify the style of char by changing files in “res/values/” and “res/layout/” figure 1 shows the char style changes in the project.



(a) (b) (c)

Figure 1: a) original app – b,c) modified app

IV. Resolution Change:

For changing the resolution, the first step is to update the SBIntervalRead and SBIntervalUpdate Seekbars to show the 6 steps instead of 4 steps in the original app by inserting the max field value of these components to 5 in the activity_main.xml. In the ActivityMain.java, by changing code regarding the event handler for Seekbars, we can access to different resolution and apply them to the program. Figure 2, 3 depicts all changes regarding the resolution in the app.

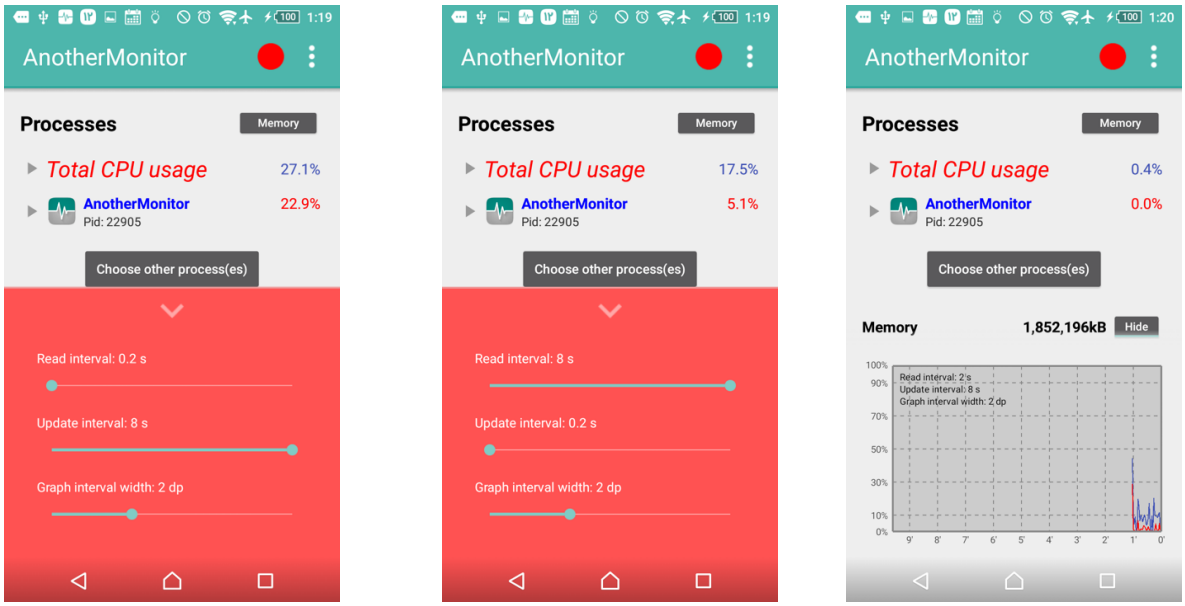


Figure 2: changes regarding the resolution in app view

```
mSBRead.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {

    }

    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {

    }

    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
        seekBar.performHapticFeedback(HapticFeedbackConstants.VIRTUAL_KEY);
        int t = 0;
        switch (mSBRead.getProgress()) {
            case 0: t = 250; break;
            case 1: t = 500; break;
            case 2: t = 1000; break;
            case 3: t = 2000; break;
            case 4: t = 4000; break;
            case 5: t = 8000;
        }
        mTVIntervalRead.setText("Read interval:" + " " + mFormatTime.format( number: t/(float)1000) + " s");
    }
});
```

Figure 3: the cod being modified regarding the resolution

V. Core Usage

A new class, CpuStat, is responsible to provide the information regarding the cores. To this end, the CpuStat parses the /proc/stat and extracts the core information by the "getCpuUsage(int cpuId)" function. For showing the results in the app, a new TextView, picturing every core usage, is added to the activity_main.xml. its information updates inside the drawRunnable thread in ActivityMain.java. Figure 4 illustrates all modifications pertaining to core usage.

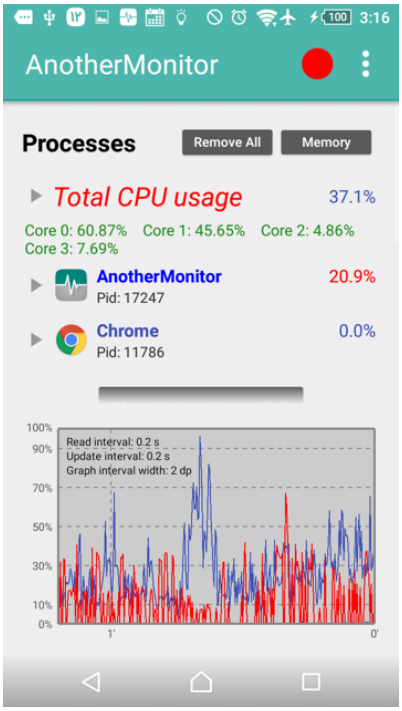


Figure 4: core usage output

VI. Reference

[1]. AnotherMonitor Source Code: https://github.com/AntonioRedondo/AnotherMonitor

[2]. AnotheMonitor Video: https://www.youtube.com/watch?v=n9P6WD7nsu4

[3]. Hardy, Brian, and Bill Phillips. *Android programming: The big nerd ranch guide*. Addison-Wesley Professional, 2013.

[4]. Android Studio Tutorial: https://www.javaworld.com/article/3340234/tutorial-series-android-studio-for-beginners.html