Improve the performance of the Android operation of external human interaction device

Naoto Ogawa
Department of Computer Science
University of Tsukuba
1-1-1 Tennodai, Tsukuba, Ibaraki, 305-8573, Japan

Shuich Oikawa
Division of Information Engineering,
Faculty of Engineering, Information and Systems
University of Tsukuba

Abstract—Smartphones become popular. They have various sensors and connection modules. They employ a mobile operating system for the control of sensors and modules. In this situation, a large amount of data comes to the operating system from sensors and modules. While such data is important for effective operations of smartphones, it is not well known that how it affects the operating system performance.

In this paper, we prepare an experiment environment that mimics a smartphone. We selected the Android as an operating system and the Pandaboard-ES as a target board. The Pandaboard-ES equips a Bluetooth module. We select Wii Remote to operate the Android via Bluetooth. We implement programs to use Wii Remote as the input device of the Android, and improve this program.

I. INTRODUCTION

A. Purpose

Recently, smart phone spread all over the world[1]. In 2008, almost all cellular phone are feature phone, but now 70% of cell phone are smartphone. Furthermore, the adoption rate is expected to reach 80% by the end of 2016.

The difference of feature phone and smartphone is that the smartphones employ a mobile operating system (OS) say iOS or Android. Another feather of the smartphone is that it has kinds of modules: GPS, Bluetooth, acceleration sensor, direction sensor, illuminance sensor and so on. Because it has many modules and it can connect kinds of external devices, many data come from modules and/or external device to the android, so android must process those data. These data is too large, so if we don’t process these data effectively, the application goes down at worst. The study about that how we can process data from the external device effectively isn’t done and the study about connecting the external device with Android isn’t also done.

The purpose of the present study is that we connect the external device with the Android and handle the Android by the external device, and we examine how the data from the external device process improve the processing time.

B. Method

We implement a program to analyze how the data from the external device is handled. This program get data via bluetooth and make keyboard input event from data. We get the each processing time by using this program.

C. Related Works

Benchmarking for Android System already exist[2], but this study mentions a CPU load of Dalvik virtual machine and analysis this CPU load. Therefore, this study is different in the point of, not for sensor device.

The study using sensors of the Android platform is present[3]. This study pays attention to the data from sensor in itself and don’t present the influence for the Android system.

The studies using Wii Remote as a human interaction device are present[4][5][6], but the platform of these studies are not Android system.

The study using the external device as a human interaction device is present[7]. This study use kinect[13]. The Kinect is enable to detect the distance and the picture by using the sensor. In this study, they make input event with human gesture.

The investigation of input interface using Acceleration sensor of the Android exists[8]. In this investigation, the data from acceleration sensor makes input event. This study is different from our study in the point of that they don’t use the external device.

II. EXECUTION ENVIRONMENT

We set up the following environment to get data from the external device and handle these data.

A. OS

We choose android[9] as an embedded operating system on smartphone because android is open source software, and it is Linux based operating system so we can use some software for Linux. The version we selected is 4.0.4(Ice Cream Sandwich).

B. Board

We choose Pandaboard-ES[10] for the target board. Pandaboard-es is a single-board computer based on the OMAP4460 developed by Texas Instruments. The features of Pandaboard-es are a dual-core 1.2GHzCPU and 384MHz GPU, Onboard 10/100 Ethernet, 802.11 b/g/n, Bluetooth v2.1, and so on. Some smartphone use the OMAP4460, so this board is similar to the smartphone. Fig.1 is the appearance of Pandaboard-ES.
C. Wii Remote

We choose the Wii Remote as an external devise to connect with the Android. Wii remote is a standard controller for Nintendo Wii[11].

This controller connects to the console by Bluetooth. We will use this controller to operate android on Pandaboard-es. Fig.2 is the appearance of Wii Remote.

III. IMPLEMENTATION

We implement a program to use Wii Remote for handling the Android. This program is based on Wiiuse. Wiiuse is a C library to connect Wii Remote by Bluetooth. This program enable to get the data from the Wii Remote and Nunchuk if connected, but this library is for Linux and Windows, so we can’t use this library on the Android originally. So, we must make a change to the library to use on the Android.

The Android NDK is a tool set that enable to make Android applications or executable program from C and/or C++ source files. We enable to use Wiiuse by using this tool and preparing the required libraries.

We use Android Debug Bridge (adb) to connect with the board and a PC. After connecting, we can use the shell of the Android shell from the PC’s terminal. We execute the Wiiuse on the shell from other PC. Fig.3 is the system configuration diagram.

Fig. 1. Pandaboard-ES  Fig. 2. Wii Remote

III. IMPLEMENTATION

We implement a program to use Wii Remote for handling the Android. This program is based on Wiiuse. Wiiuse is a C library to connect Wii Remote by Bluetooth. This program enable to get the data from the Wii Remote and Nunchuk if connected, but this library is for Linux and Windows, so we can’t use this library on the Android originally. So, we must make a change to the library to use on the Android.

The Android NDK is a tool set that enable to make Android applications or executable program from C and/or C++ source files. We enable to use Wiiuse by using this tool and preparing the required libraries.

We use Android Debug Bridge (adb) to connect with the board and a PC. After connecting, we can use the shell of the Android shell from the PC’s terminal. We execute the Wiiuse on the shell from other PC. Fig.3 is the system configuration diagram.

We want to use Wii Remote to handle the Android, but Wiiuse supply connect function with Wii Remote and get data from the Wii Remote, so we must implement input function to the Android. Wii Remote’s each cursor key corresponds to the keyboard’s each cursor key. TABLE I is the table from the other Wii Remote’s button to the keyboard.

We implement input method in two ways. First method use shell command and second method use Linux libraries.

A. Shell command

Android shell has input command. This command makes keyboard input event. This command uses like this

```bash
# input keyevent EVENT_CODE
```

These EVENT_CODEs are defined in Android’s source code.

We duplicate the process. Duplicated process execute the input command. This commands generate Virtual Machine(VM) and execute the java programs on the VM. This java program makes key input event and sends key input event to the Android Application. The other process waits for the first process completes. One input process is completed now. This program polls till next input comes. Fig.4 shows the summary of this program.

We measure the time to complete one key input event on each version. We use gettimeofday function to get time. We get time when we push the button of Wii Remote. This is start time. We get time when the input event is completed. This is end time. We determine that the value obtained by subtracting the start time from end time is the execution time.

<table>
<thead>
<tr>
<th>Wii Remote Correspond Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wii Remote</td>
</tr>
<tr>
<td>HOME</td>
</tr>
<tr>
<td>A</td>
</tr>
</tbody>
</table>

B. Linux libraries

We implement another version. This version makes virtual input device. Android prepares User Input module, so we use this module. We open uinput[14] device file to use this module and write device type we want to generate to this file. In this case, the type is keyboard.

We can generate input event by writing the relevant value to the virtual input device file descriptor. When this program gets data from Wii Remote, this program makes key input event by writing the virtual input device file descriptor. One input process is completed now. Fig.5 shows the summary of this program.

IV. RESULT

We measure the time to complete one key input event on each version. We use gettimeofday function to get time. We get time when we push the button of Wii Remote. This is start time. We get time when the input event is completed. This is end time. We determine that the value obtained by subtracting the start time from end time is the execution time.
Fig. 5. use ioctl function

Fig. 6. compare two versions

Fig.6 is the graph to compare the execution time of two versions. This graph is logarithmic graph.

The execution time of the input version is about ten thousand times the execution time of the ioctl version at all buttons. The input version is too late so when we use Wii Remote to handle the Android and hit buttons repeatedly, the application doesn’t work as we mentioned immediately. We get stressed in this situation.

On the other hand, the ioctl version is so fast, and all execution time are under the one millisecond. We don’t feel stress when we use the Wii Remote as an input device.

V. DISCUSSION

We implement the method to handle data from an external device and measure the execution time at each method. Significant difference appears in the execution time. The method that use input command is so late that it is not practical. On the other hand, the method that use ioctl is so fast that it can use input method. This result is as we expected because the method that use ioctl is a single process and runs on the real machine, whereas the method that use input command duplicate the process and generate VM. Therefore, we can say that it is obvious that the method using ioctl is faster than the method using input command. Furthermore, we consider that when you connect an external device to the Android and handle data from the device, you should implement the program to handle the data with C and/or C++.

We estimate that the Android increasingly spread from now on and the demand for connecting the external device with the Android increase.

VI. CONCLUSION

A. Summary

In this paper, we describe that how the data from an external device is handled. To get the data from the external device, we connect the Android with the Wii Remote with Wiiuse library. We implement input method in two ways. First one makes another process in the program, and use shell function to make input event. Second one use C library to make keyboard input event from the data of the Wii Remote. Comparing two methods, the second one use short time to complete a keyboard input event. At that time, we consider that our study is instrumental.

B. Future works

We think that our implementation has three points that should be improved.

1) We use C library to make keyboard input event. This program runs on the user space, so we think that many context switching will happen between user space and kernel space. We expect to implement device driver for the Wii Remote not to happen many context switching.

2) When we want to connect another external device to the Android, we must prepare the program that dedicated to handling data from the device. It is inconvenient so we must develop the system that enable us to connect the external device with the Android.

3) The program we implemented needs shell to execute, so we must implement interface with java using the JNI. If so, we can use the program from Android applications.

REFERENCES

[1] Sangwon Lee and Senmi Lee, Diffusion of Smartphone in Global Telecommunication Markets