Research on Operating System Migration Method Based on Domestic Embedded Devices

Yan Li¹, Shiran Zhang¹, Jian Li², Haikuo Xi¹, Tianying Dong¹, Zexu Du³*

¹State Grid Chengde Power Supply Company, Chengde, Hebei, China
²State Grid Jibei Electric Power Co.Ltd, Chengde, Hebei, China
³Artificial Intelligence on Electric Power System State Grid Corporation Joint Laboratory (GEIRI), Global Energy Interconnection Research Institute co.Ltd, Beijing, China
*email: duzexu@geiri.sgcc.com.cn

Abstract. At present, Google's Android system has the characteristics of open source, rich software functions, and good user experience, and has been highly recognized by a wide range of applications and the market. If the Loongson platform wants to make a difference in mobile terminals and other fields, it needs to support the Android system. However, although the native Android system supports multiple architectures, it only performs a lot of optimizations for the ARM architecture. For other architectures, Google only provides a generic version. This brings a huge challenge to the Loongson processor based on the MIPS architecture. Based on the analysis of the Android system, this paper studies the Android transplantation method of the Loongson 2K1000 platform in detail realizes the transplantation of the Android Lollipop system on the platform and has passed the verification of the Loongson 2K1000 platform.

1. Introduction

The Loongson series processors are general-purpose processors based on the MIPS instruction set and independently developed by China with completely independent intellectual property rights. The Loongson 2K1000 processor integrates two GS264 processor cores on-chip and has a wealth of chip peripheral interfaces. It is designed for network security, mobile intelligent terminals, high-end embedded applications, industrial control, Internet of Things and other fields, and has broad market prospects.

Currently, the Loongson platform mainly uses Linux customized distributions. However, in the fields of mobile smart terminals, the Android[1] operating system developed by Google has the characteristics of open source, rich software functions, and good user experience. With the exception of Apple, almost all relevant domestic and foreign manufacturers support the Android system. In 2017, its market share reached 86%[2]. Therefore, it has become an inevitable choice for the Loongson 2K1000 platform to support the Android operating system.

Based on the analysis of the Android system and the architecture of the Loongson 2K1000 platform, this article has deeply studied the key technologies of Android transplantation on the Loongson 2K1000 platform, and finally realized the complete operation of the Android Lollipop system on the platform.
2. Related work

2.1. Android system architecture
Android is an open-source operating system based on the Linux kernel that was released by Google at the end of 2007 for mobile terminals. The latest version of the Android system has been iterated to 9. Its system architecture includes Linux kernel, hardware abstraction layer, system runtime library and Android runtime environment, application framework layer and application programs.

The Android system runs on a specially adapted Linux kernel. It provides core services such as process management, memory management, security management based on user permissions, and the underlying hardware driver model for the upper Android system, while shielding most of the underlying hardware Specific differences[3].

The Android system has done a lot of tailoring to the commonly used glibc library on the Linux operating system and the standard tool chain of the Linux platform. At the same time, it also adds a number of new modules to the Linux kernel to optimize the performance of the Linux kernel in related aspects. Adapt to the application scenarios of the Android system[4]. These modules include the Binder driver for inter-process communication in the Android system, the Ashmem driver for the anonymous shared memory mechanism, the Logger driver for the lightweight logging system, and the Low Memory Killer for process lifecycle management. Mechanism etc.

The Android system architecture uses a layered design method[5], and each layer provides a unified service for the upper layer to shield the differences between the layers. The functions of each level are as follows:

- The hardware abstraction layer is a layer between the Android system in the user space and the Linux driver in the kernel space. Its purpose is to abstract the hardware, hide the hardware interface details of a specific platform to the upper layer, have hardware independence, and facilitate the transplantation of multiple platforms.
- The Android local shared function library provides the most basic local operating environment and local function calls for the application layer Java programs and the Dalvik/Art virtual machine in the Java runtime environment.
- Both the application layer and application framework layer of the Android system use Java language as the development language. The Android system has specially developed the Dalvik virtual machine and the Art virtual machine used after Android 5.0 to execute upper-level Java programs.
- The application framework layer provides a management program that calls the core programs within the system. The application layer implements a series of core applications, such as browsers, notebooks, calendars, maps, etc. The programs can be implemented by calling local codes through the JNI mechanism.

2.2. Loongson 2K1000 platform
The Loongson 2K1000 processor is a general-purpose RISC processor with a 40nm process, which integrates two GS264 processor cores compatible with the MIPS64 R2 instruction set. Its working frequency is 1GHz, power consumption is 1 to 5W, and it integrates GPU, two PCIE2.0, one SATA2.0, four USB2.0, two DVO, 64-bit DDR2/3 and other multiple functional modules on-chip. Table 1 shows the Loongson 2K1000 processor platform configuration [6].

| Name   | Characteristic                          |
|--------|----------------------------------------|
| CPU    | Loongson 2K1000(GS264 x2, 1GHz, 1MB level 2 cache) |
| GPU    | Support OpenGL ES 2.0, OpenGL ES 1.1 |
| Mem    | 8GB(DDR3-1600MHz)                      |
3. System migration

3.1. Build a cross-compilation environment

According to the requirements of the Android open-source project, install the Repo tool in the Ubuntu 16.04 (64-bit) environment and obtain the Android Lollipop source code. Select the software toolkit that is compatible with the Android Lollipop version, and ensure that the system is installed with tools such as Python 2.7, Open-JDK 7, and make 3.82. The Android source code comes with a MIPS cross-compilation tool chain. By setting the relevant script file, you can specify that the Loongson 2K1000 processor is an implementation of the MIPS architecture. The gcc configuration compile option is -march=mips32r2 or -march=mips64r2 to compile 32-bit or 64-bit Android system.

The Linux kernel and its cross-compilation tool chain are obtained from the Loongson official open source community, and the Linux kernel can be compiled according to the relevant instructions.

3.2. Linux kernel porting

The Linux kernel is obtained from the Linux 3.10 version of the Loongson official open-source community. This version contains all the drivers required by the Android system on the basis of supporting the Loongson 2K1000. The Loongson CPU core only supports 16K page size by default, while the Android system only supports 4K page size. If the Android system is changed to support 16K pages, it will not only have a huge workload but also make the system easy to crash. In summary, setting the Linux kernel page table size to 4K is a better choice. Then select all the Android drivers under Device Drivers->Staging drivers-->Android in make menuconfig, including Binder driver, Ashmem driver, Logger driver, Low Memory Killer component, etc.

3.3. Android system porting

According to the above detailed analysis of the Android system framework, the application layer is designed in Java language and does not need to be modified. Therefore, the transplantation of the Android system is mainly aimed at the transplantation of the hardware abstraction layer, the local shared library, the Dalvik/Art virtual machine and the application framework layer.

The Android source code directory structure is mainly composed of the following parts[7]: Makefile (global compilation script), Art/Dalvik (Java virtual machine), bionic (basic library code), bootable (boot and recovery program), build (compilation and Configuration script), development (program development simulator and tools), device (platform-related configuration files), external (system external library), frameworks (application framework), hardware (hardware platform-related), packages (applications), prebuilt (Android compilation tool chain), system (Android system library).

First, create the boardConfig.mk, device.mk and other compiled configuration files of the 2K1000 platform in the device directory. These files mainly describe the module-level information of the target platform, such as the configuration information of the GPU, WIFI module and other devices. When the compilation tool compiles these device-related modules, it will look for the corresponding source file and process the pre-compiled information in the source code according to the parameters described in the above configuration file.

If the compiler option of the cross-compilation tool is -march=mips32r2, then the compiled Android system is in 32-bit mode. 2K1000 supports floating-point co-processor, there are 32 floating-point registers $f0-$f31, actually each floating-point register is 64 bits. However, according to the MIPS32 convention, these 32 floating-point registers are 32 bits, so you can only use even-numbered floating-point registers, because when doing double-precision floating-point operations, even-numbered floating-point registers and odd-numbered floating-point registers will be combined into one 64-bit registers are used to store 64-bit floating-point numbers, so odd-numbered floating-point registers cannot be used alone[8]. To solve this problem, it is necessary to modify the relevant assembly code of the odd floating-point registers used in the Art virtual machine. For example, the
code is $f0, 0($a1); $f1, 4($a1); uses $f0 and $f1 floating-point registers, if you are compiling a 32-bit system, then these two codes will be replaced. It is the macro assembly code LDu $f0, $f1, 0*8 $a1, $t1. The prototype of this macro assembly code is LDu feven, fodd, disp, base, temp, this macro assembly expands to: is $f1, $disp($base); lw $t1, $disp+4($base); mthc1 $t1, $feven. The function of this macro assembly code is to store the first 32 bits of data in the memory in the lower 32 bits of the $f0 register, and then store the last 32 bits of data in the memory in the upper 32 bits of the $f0 register. In this way, the use of odd floating-point registers can be avoided.

3.4. Android graphics subsystem refactoring

If the Android system simply uses a general-purpose processor to be responsible for the calculation, rendering, and drawing of all 2D/3D graphics in the system, its fluency is far from being able to meet the requirements of normal use. Therefore, almost all Android devices are equipped with a hardware graphics processing unit (GPU) to process the computing tasks of the graphics subsystem in the operating system. Loongson 2K1000 is also equipped with a graphics processing unit, which supports OpenGL ES 2.0 and OpenGL ES 1.1 graphics program interfaces[9].

In the Android graphics subsystem[10], the Surfaceflinger service provides services such as drawing stream synthesis, graphics rendering, and cache management to the application, and the implementation of these services requires calling the OpenGL interface and EGL interface provided in the Android local shared library. The architecture of the Android graphics subsystem is shown in Figure 1. The subsystem can be realized by pure software simulation. After proper configuration, the libGLES_android.so dynamic library is compiled. The library is written in C++ language and does not rely on any graphics processing hardware modules, and has the commonality of each target platform. However, after testing, it is found that after successfully porting the Android system to the Loongson 2K1000 platform, the pure software graphics subsystem can make the system work normally, but its performance cannot meet the requirements of smooth system operation, and the graphics interface will frequently show cards that users cannot tolerate. Sensation.

In the original graphics subsystem of the Android system, the graphics hardware drivers included in the hardware accelerated graphics library are generally provided by graphics hardware manufacturers in a closed source form, so the Loongson 2K1000 cannot enable graphics hardware acceleration under the original Android graphics subsystem. However, the Mesa 3D graphics library[11] has been ported to the operating systems on each Loongson platform to achieve hardware acceleration of graphics processing. The Android x86 project[12] also uses the Mesa 3D library to achieve graphics hardware acceleration. Therefore, the use of Mesa 3D library can realize the Android
graphics hardware acceleration of Loongson 2K1000, and the reconstructed graphics subsystem is composed of several libraries such as libGLES_mesa.so, libdrm.so, libdrm_radeon.so and so on. The libGLES_mesa.so library is the adaptation layer of the OpenGL interface and the EGL interface, which provides the application layer with the call of the OpenGL interface; the libdrm.so library is the implementation of the direct rendering manager, and provides the hardware abstraction of the graphics processing hardware for the upper interface adaptation layer, the library depends on the user-mode graphics hardware driver, corresponding to the libdrm_radeon.so driver. The above modules constitute the user space part of the Android graphics subsystem. In addition, the DRM driver and GPU driver in the kernel space are required to cooperate with them to realize the Android graphics subsystem of the Loongson 2K1000. The reconstructed graphics subsystem is shown in Figure 2.

4. Migration results and performance analysis
After completing the transplantation of the Android system according to the previous article, it is compiled by the cross-compilation tool chain to generate an executable system for the Loongson 2K1000 platform. Figure 3 shows the running effect of the Android Lollipop system on the Loongson 2K1000 platform. Due to the limitation of the peripheral hardware equipment configuration of the 2K1000 platform, some system functions such as making phone calls, connecting to WIFI, and taking pictures are limited. The basic functions of the rest of the system can be used normally, and the system interface is smooth, and the operation response is timely.
0xBenchmark is an open source performance testing tool specifically for the Android system[13]. This tool is used to test the arithmetic performance of the system. The test data is shown in Table 2. After enabling graphics hardware acceleration, the 2D/3D graphics performance, the test data is shown in Table 3, and Figure 4 is the test running scene record. As shown in the test results, the transplanted system already has a certain practicability.

| Tab. 2 Loongson 2K1000 Android system computing performance test results |
|---------------------------------------------------------------------------------------------------|
| **Benchmark** | **Test Case** | **Result (Mflops/s)** |
| Linpack | Linpack for Android | 3.89 |
| Scimark2 | Composite | 5.04 |
| | Fast Fourier Transform | 3.58 |
| | Jacobi Successive Over-relaxation | 9.78 |
| | Monto Carlo integration | 0.36 |
| | Sparse matrix multiply | 5.61 |
| | Dense Lu matrix factorization | 5.88 |

| Tab. 3 Loongson 2K1000 Android system graphics performance test results |
|---------------------------------------------------------------------------------------------------|
| **Benchmark** | **Test Case** | **Frame rate (fps)** |
| 2D | Canvas | 29.90 |
| | Circle | 22.74 |
| | Circle2 | 19.77 |
| | Rect | 6.23 |
| | Arc | 13.86 |
| | Image | 3.85 |
| | Text | 56.59 |
| 3D | Cube | 33.96 |
| | Blending | 61.09 |
| | Fog | 60.60 |
| | Teapot | 35.08 |

Fig. 4 Test run scenario record
5. Conclusion
This article is mainly based on the processor characteristics of the Loongson 2K1000 platform. The source code of the Android Lollipop system has been transplanted and optimized, and the Android graphics subsystem has been reconstructed. The system can enable graphics hardware acceleration. Makes the transplanted system has a certain practicability. The next step will focus on the research on the performance optimization of the Loongson platform Android system and the binary compatibility of Android applications.

Acknowledgments
This work is supported by State Grid Chengde Power Supply Company Science and Technology Research Project (Deepen the application of artificial intelligence technology for power transformation (distribution)).

References
[1] Gargenta, M 2011 Learning android. “O’Reilly Media, Inc.”
[2] Zhao X 2014 Malicious application behavior analysis and detection on Android platform. Beijing Jiaotong University
[3] Han C and Liang Q 2011 Android system-level in-depth development: porting and debugging Electronic Industry Press
[4] Yang H M, Zhang T, Zhao M, You J and Dong M 2014 Automatic adaptation technology for BSP transplantation based on Android. Computer Engineering, doi: 10.3969/j.issn.1000-3428 049
[5] Lei M, Jin T, Xu XL and Qi J 2012 Research on Android system transplantation for Loongson platform Computer Engineering and Applications 48(22) pp 70-73
[6] Huiyan, L, et al 2019 "System Design of Wireless Meteorological Element Acquisition Based on Loongson 2K 1000." International Conference on Meteorology Observations (ICMO) IEEE
[7] Luo SY 2012 Scenario analysis of Android system source code Electronic Industry Press
[8] Sweetman 2010 Dominic. See MIPS run. Elsevier
[9] Sellers, Graham, Richard S 2013 Wright Jr, and Nicholas Haemel. OpenGL superBible: comprehensive tutorial and reference. Addison-Wesley
[10] Gunasekera, S 2012 Android Architecture. Android Apps Security CA pp 1-12
[11] Paul B 1999 The mesa 3d-graphic library. http://www. mesa3d. org
[12] Krajci, I and Darren C 2013 Android on x86: An Introduction to Optimizing for Intel® Architecture. Springer Nature
[13] Chang, K, et al 2014 “Memory behavior profiler for Android applications” 2014 IEEE 3rd Global Conference on Consumer Electronics (GCCE) IEEE