Paravirtualization for the ARM MP-Core using the L4 Fiasco (ARM version) with the Linux/Android stack

Prasanna Kumar Krishnamurthy (pkrishn6@ncsu.edu)
Varun Ganesh (vganesh2@ncsu.edu)

Abstract:

The goal of the project was to successfully complete the paravirtualization of Android and be able to load a virtualized Android over the hypervisor L4 on the ARM PB11MPCore Development board. The kernel image is to be obtained from a TFTP server and the Android root filesystem is to be loaded over NFS (using a ROOT NFS mount). The project required understanding and continuing the existing work [1] done in trying to bring up L4Android.

Introduction:

Mobile virtualization platforms typically attempt to isolate computing resources from applications/platforms that use them. This isolation provides a means to regulate the access that the applications have to the computational resources, thus enabling malware detection capabilities. The isolation is achieved by inserting a virtual hypervisor layer between the operating system and the hardware. A guest operating system that runs on top of the hypervisor layer can access the underlying hardware via paravirtualized system call only. This enables the hypervisor layer to govern all the interactions that take place between operating systems (and the layers above it) and the hardware. Many policies and checks can be implemented in the hypervisor layer to ensure that unauthorized access and data transmission are thwarted, thus ensuring that the systems in question are not compromised.

Native L4Linux that runs over L4 cannot be used to load the Android filesystem. This is due to the inherent differences between Linux and Android kernel[2] such as kernel enhancements in Android that include alarm driver, ashmem (Android shared memory driver), binder driver(Inter-Process Communication Interface), power management, low memory killer, kernel debugger and logger modules. Thus an android kernel needs to be modified by changing all sensitive instructions (ones that don't trap) to hyper-calls, thus making it the Android counter-part of L4Linux. One implementation of such a kernel termed L4Android has been developed by Operating Systems group at Dresden University [3].

Steps Accomplished:

1. We have compiled the L4 kernel (fiasco) along with L4Re and have been able to compile and build elf image from L4Linux (for PB11MPCORE) required as a base for the Android stack. We have also been able to compile L4Android with the same configuration as L4Linux and the Android file-system Froyo for PB11MPCORE (for ARMv6k instruction set)

2. We have been able to bring up the development board, configure the baud-rate settings on the PB11MP boot monitor and load U-Boot. We have been able to load the existing L4Android image on the SDCard and identify the point of issue at init. We have been able to partition the SDCard to load the compiled L4Linux image and continue to burn it to the flash memory.

3. We have been able to setup tftp server at a public IP address and configure the U-Boot environment variables to boot image from the tftpserver. The bootstrap_L4Linux_ARM.elf image was obtained by doing a mkimage combining the L4, L4re and vmlinuz.arm from L4Linux. We were then able to obtain the image on the board through tftp and after booting, obtained command line prompt containing the ramdisk directories

4. We have been able to identify virtual bus configuration in the L4 kernel to provide L4Linux with an Ethernet interface in the virtualized environment. This required configuring the module list to include the device tree blob for L4Linux to identify the devices present, and also include the SMC911x driver in the L4Linux kernel during the compilation phase to successfully acquire the device during Device Scan and to pass through the device to the virtual bus created by L4 to L4Linux.

5. We have identified the boot process required for NFS as follows
   a. The kernel image is loaded through the tftpboot and written to the memory.
   b. The NFS environment variables at U-Boot are set to point to the NFS server containing the Root filesystem required by the kernel. For L4Linux, this is to be done in l4lx.cfg
   c. When the kernel-image is booted, it acquires the root filesystem through the NFS server.

   We have been able to configure the NFS server to allow access to the development board and have been able to build an NFSRoot filesystem as required by the kernel (as mentioned in step 1)

6. We were able to replace the L4Linux vmlinuz.arm image with L4Android's vmlinuz.arm and get it to boot on the board. By changing the boot configuration script loaded by L4, we were able to pass RootNFS boot command to L4Linux. We have configured a static IP and default gateway to the board but configuring it as DHCP is simpler.

   Now, on bootup, L4Android sends a MNT request to the NFS server. The init.rc file in the NFS root directory in the server can be used for configuring what directories need to be included.

   Currently, L4Android boots up on PB11MPCORE and successfully mounts the root filesystem over NFS but init fails without any explicit error notification. We had tried possible solutions for resolving the above error (such as incorrect dynamic linking of /lib setup [4]), but as of yet they have been of no avail.

Issues resolved:

1. Individual downloads of L4, L4re and L4Linux have boot problems and clear steps [5] as to what needs to be followed when doing a make image combining the three were not given. This had to be solved by using the snapshot image at L4re downloads and the lower level configuration details are abstracted.

2. The original source tree maintained at NCSU CSC repository currently is not configured to include the virtual bus and the L4Linux version also doesn't have a SMC911x option in the make menuconfig.

   The same problem existed with L4 snapshot download. The SMC911x menuconfig option was present only in the individually downloaded L4Linux image. This was identified and solved by initially doing a make of the snapshot image and then substituting the vmlinuz.arm with the newly compiled one from the individual image.

3. Baud-rate issues existed when the boot was attempted. This was because of the fact that U-Boot was configured for 38400 baud whereas the L4 image worked at 115200 baud.

   As U-Boot was being run over ARM Bootloader as a second stage, configuring environment variables [6] for U-Boot does not work for the current version installed on the board (2009.03) [7]. A work around was to change the baud rate of minicom when booting L4.

4. The initial boot-prompt froze on bootup. This was fixed by adding a serial_esc line in modules.list while compiling the image.

5. L4 when made to run over PB11MPCORE stalls when initializing memory due to synchronization issues. A work-around for this was to add printf statements during initialization in the bootstrap server code (under regions.cc).

Deliverables:

1. Bootable and working elf image compiled from L4, L4re and L4Android configured to load Filesystem from NFS.
2. Source tree for L4Android and L4 snapshot from which the image in step 1 was created
3. README document containing elaborate steps on how to obtain the source tree in step 2, configuration details (creating l4lx.cfg, l4lx-vbus.io, modules.list etc.), changes to be made to L4 and bringing up the NFS server
4. Steps for compiling Android filesystem [8]

Conclusion:

We were able to successfully configure and compile images for L4Android, L4 and L4re and configure L4Android to do an NFS mount. Currently, we are facing issues with init during boot-up (Init is called and fails). This issue has been identified by the previous work on the board but the resolution is unknown.

Project webpage:
http://www4.ncsu.edu/~pkrishn6/CSC714/first_page.htm

References:

[1] Exploring Virtualization Platforms for ARM-based Mobile Android Devices - Thesis report by Rahul Ramasubramanian
[2] A Survey on Android vs. Linux - *Frank Maker* and *Yu-Hsuan Chan*
[3] http://www.l4android.org
[4] http://tldp.org/HOWTO/NFS-Root.html
[5] http://wiki.tudos.org/Quickstart
[6] http://www.embedian.com/index.php?main_page=ubootev
[7] http://lists.denx.de/pipermail/u-boot/2011-January/085350.html
[8] http://www.linux-arm.org/LinuxKernel/LinuxAndroidPlatform