VMXLINUX: IMPROVING SHIM LAYER PORTABILITY

Petre EFTIME[1], Mihai CARABAS[2], Lucian MOGOSANU[3],
Laura GHEORGHE[4], Razvan DEACONESCU[5]

This paper presents a solution for reducing the time it takes to update the virtualization layer required for running Linux on top of an L4 microkernel as a virtual machine. One needs to make significant changes in the Linux kernel to allow it to run as a virtual machine due to the early support of virtualization extensions in embedded hardware platforms. This is essential for being able to keep up with the development of the Linux kernel and running the latest versions as virtual machines on new devices.

Keywords: Virtualization, ARM, L4, Linux, portability

1. Introduction

VMXLinux is a virtualized Linux kernel running on top of VMXL4, a security oriented microkernel, part of the L4 family of microkernels. VMXL4 and VMXLinux were built to provide safety and performance for virtualization and bare-metal applications. The focus of the project is currently on the dual Android scenario - a mobile device running two virtualized versions of Android: one with strict security requirements and another without such requirements - and automotive scenarios. The flavor of virtualization used is often called *paravirtualization* (modifying the kernel in order to run on top of a mediator – in this case the VMXL4).

VMXLinux is based on Wombat Linux [5], a paravirtualized Linux kernel running on top of L4Ka::Pistachio and the Iguana runtime environment. VMXLinux replaces the parts of the Linux kernel which require privileged instructions with system calls to VMXL4. These parts are called the shim layer [9], and have been located in the `arch/l4` folder, a model inherited

[1] Master student, Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, e-mail: petre.eftime@cti.pub.ro
[2] Teaching Assistant, Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, e-mail: mihai.carabas@cs.pub.ro
[3] Teaching Assistant, Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, e-mail: lucian.mogosanu@cs.pub.ro
[4] Lecturer, Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, e-mail: laura.gheorghe@cs.pub.ro
[5] Lecturer, Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, e-mail: razvan.deaconescu@cs.pub.ro

from Wombat Linux. However, this model creates some problems when updating and porting the paravirtualization layer on a new kernel version.

The purpose of this project is redesigning the shim layer architecture to reduce the overhead of performing these operations. The proposed solution is to remove the `arch/l4` folder and create a series of patches which can be applied on top of the native architectures, which in this case is ARM. The new shim layer seems promising in this regard, with little overhead, as discussed in Section 4 and Section 5. The virtualization features brought by these patches are the subject of another project widely described in the theses [8] and [9].

The paper is structured as follows: Section 2 presents the background and related work, Section 3 presents the reasons and design of the work done to improve portability of VMXLinux, Section 3 describes the process and verification for the design, Section 5 explores how the implementation was evaluated and quantifies the results and Section 6 discusses the realization of the goal and enumerates some of the possible future improvements.

2. Related work

Virtualization is a method for running multiple isolated operating systems on a single hardware platform. It is meant to help improve security and reduce hardware cost. It is widely used in servers and platform as a service solutions. Virtualization, by definition, requires performance similar to the system on which it runs [7], but without hardware support or a virtualizable architecture this sort of performance is difficult to obtain.

ARM CPUs are not virtualizable: not all privileged instructions generate traps when executed from a non-privileged context [7]. Recently, there have been additions to the ARM architecture - ARMv7 Virtualization Extensions and ARMv8 EL2 - which allow hardware assisted virtualization on ARM based devices. However, hardware platforms with such CPUs (Cortex-A15 and Cortex-A50 series) are currently only available on high-end devices.

Paravirtualization is a model of virtualization in which the guest OS kernel is modified to communicate with the host operating system. Paravirtualization can provide better performance than other virtualization methods such as emulation or full virtualization, which must rely on runtime binary translation when hardware assisted virtualization is not possible. A number of projects exists that aim to bring high performance paravirtualization to ARM devices which lack hardware assisted virtualization.

Wombat Linux [5] is a paravirtualized Linux kernel running on top of the L4Ka::Pistachio microkernel and an Iguana runtime environment. Wombat aims to create a secure Linux environment which runs alongside other native applications. To achieve this, the Wombat kernel runs unprivileged and acts as a

server for the Linux programs. Linux itself runs as a single-threaded process on top of the Pistachio microkernel and the microkernel mediates system calls to the Linux kernel as IPCs. This approach to paravirtualization is also used by L4Linux/L4Android [4], which runs on top of the Fiasco.OC microkernel with an L4Re runtime environment. KVM for ARM [2] also uses paravirtualization, the approach being different. Compilers do not generate privileged instructions, so the KVM for ARM project replaced these instructions with calls to the KVM hypervisor. The KVM hypervisor will recognize the instructions, will execute the required operations and set the registers to the correct state before returning control to Linux. This approach is called light paravirtualization.

Xen also has two ARM variants, one of which uses paravirtualization [3], but has been deprecated in favor of hardware assisted virtualization. Hardware assisted virtualization will be preferred as more ARM CPUs will have virtualization extensions, but for now it is limited to servers and high-end smartphones and tablets.

VMXLinux is a paravirtualized Linux which runs on top of the VMXL4 microkernel. It uses a similar architecture and design as Wombat Linux [5] but has some additional features. VMXLinux is multithreaded, with specialized threads for syscall handling, interrupt handling, device mapping and fault handling for the paravirtualized kernel. Another notable addition is a series of high availability features such as process restarting and checkpointing. Multiple instances of VMXLinux can run on top of the VMXL4 microkernel.

3. Shim Layer Redesign

VMXLinux is a paravirtualized Linux kernel which runs on top of L4 microkernels. This model of virtualization was chosen because the ARM architecture, which is used by a majority of smartphone and tablet devices, is not virtualizable and while there are ARM CPUs which have virtualization extensions, as stated above it is not a common feature for now. VMXLinux can run alongside other bare metal applications or virtualized Linux kernels and can have direct hardware access.

In the case of VMXLinux, the code required for paravirtualization is called the shim layer. The shim layer replaces privileged instructions required for memory management and thread management, and the interrupt controller to fit in with the VMXL4 microkernel. Communication between programs and the Linux kernel is also done via inter process communication (IPC) rather than software interrupts.

The shim layer model of VMXLinux is inherited from Wombat Linux and adds a new code structure to the Linux kernel named L4 as shown in Figure 1 (this new architecture is also referred as `arch/l4` in this paper). Because

Wombat ran on multiple architectures (x86, ARM and MIPS), this layout allows the existence of common files and architecture specific files [5], however, it also fragments the code, which makes upgrades to the Linux kernel difficult, as modifications have to be split up and possible inconsistencies between architectures have to be fixed. Also, the Kconfig and Kbuild systems, needed for configuring and building the Linux kernel have to be modified to work with the L4 architecture. These issues made updating the Linux kernel and adding new hardware platforms a lengthy and difficult process [1][8].

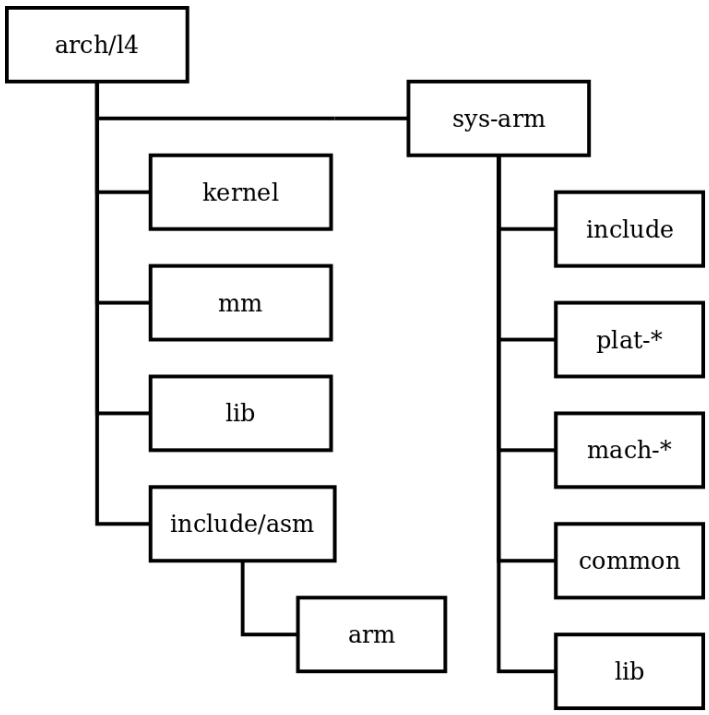

Fig. 1. `arch/l4` code structure

A new shim layer model without multi architecture support (see Figure 2), based by adding a series of files and patches on top the original architectures would resolve the issues regarding process length and Kconfig/Kbuild functionality. Most of the L4-specific code is condensed in a few files which can be chosen at build time instead of the original architecture specific files, and other mostly minor modifications would be made to a few other files. The issues with this approach are the fact that support for multiple architectures would require for each architecture to be modified and synchronization between architectures might be problematic. In addition bugs might be harder to locate since L4 code will be mixed with architecture code.

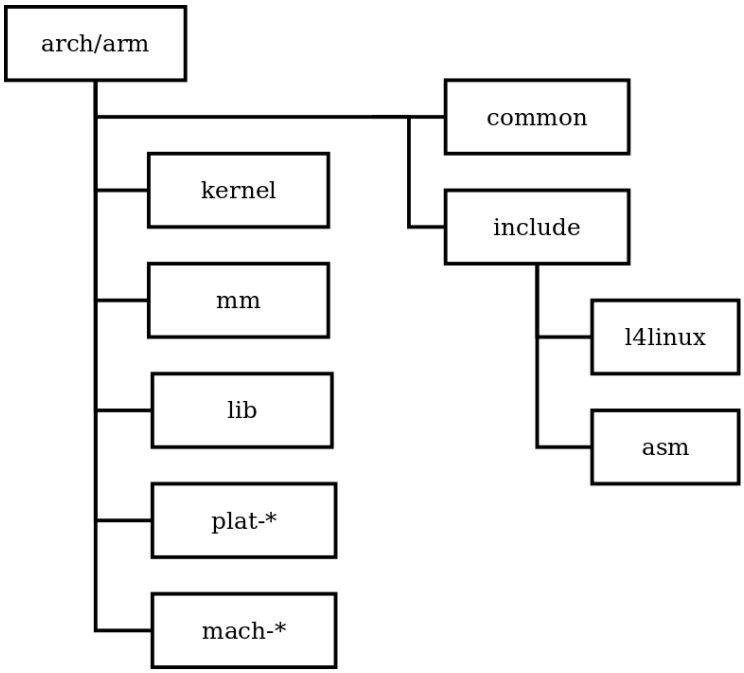

Fig. 2. `arch/arm` code structure

4. Implementation

We made two steps in order to implement the redesign and validate it: the first one involved merging the L4 architecture with the ARM architecture, on top of the 3.0.8 version of the Linux kernel, and the second one was to patch the ARM architecture of the 3.4.0 kernel with the modifications made to the ARM architecture of the 3.0.8 kernel.

4.1. Porting arch/l4 to arch/arm

This step was lengthy and it involved studying the differences between the two architectures and identifying the differences and the cause of the differences (see Figure 1 vs. Figure 2). The causes could be: the function was specific to `arch/l4`, the function was from an older kernel version **(**`arch/l4` was not completely up to date) or the function contained some `arch/l4` code. Although this approach helped limit some of the issues, by identifying most of the common and different code, the process was not easy, as the layout of the two architectures was slightly different, which lead to compiling and linking issues during the porting process.

A partially automatic merging was tried, identifying macros which were different between architectures, however, many macros had multiple definitions and the reason for which they were different was not always immediately obvious. This ended up being a dead end as it was difficult to find the relevant macros and it did not take functions into consideration at all. The merging had to be done manually.

The resulting architecture has a very similar structure with the original ARM architecture. There are a few files added: files of the form `filename_l4.c`, which replace the functionality of the original file, `filename.c`, files with L4 specific functionality and a set of headers found in `include/l4linux`. These files account for around 15000 lines of code. In addition to these files, many files were modified, especially header files. These modifications add up to around 3500 lines of code and include drivers. The porting process ended when the Linux Test Project (LTP [10]) returned the same results for both the `arch/l4` architecture and the paravirtualized `arch/arm` architecture.

The immediate positive result of this port was being able to use the Kconfig system, which makes configuration of the kernel easier, and could allow for multiple devices being supported by the same Linux kernel. This was not really an option previously as configuration was difficult and had to be done by editing the configuration files.

4.2. Paravirtualization of the 3.4.0 Linux kernel

To test that the new shim layer was easily upgradable, we ported the virtualization layer to the 3.4.0 Linux kernel. The process was to extract the modifications from the 3.0.8 kernel into two patches: one with all the added files and another with all the modifications brought to already existing files. The patch with the added files was applied immediately, but the second patch failed. However, it is likely that applying patches to consecutive versions of the kernel would work at least partially. The modified lines of code had to be applied manually, but this process only took one or two person days. A running kernel was obtained in about eight to ten person days worth of work.

The results of the LTP were very similar to the ones obtained from Linux 3.0.8 (and somewhat better than the native version, however, the LTP used was slightly out of date). This meant that the port was successful and the overhead for upgrade was minimal at best. While not all of the new Linux configuration options are available, the basic functionality is available and most the `arch/arm` port will be very useful for keeping the VMXLinux synchronized with the mainline Linux branch.

5. Evaluation

The paravirtualized shim layer had to be validated. A good start is getting the kernel to finish setting up and start the init process, which indicates that the hardware could be initialized, the file systems mounted and an userspace process was able to run and utilize at least a subset of system call. The second step is testing the rest of the environment and system calls.

A standardized test for Linux systems is Linux Test Project (LTP [10]). This project contains a number of regression and conformance tests for the kernel and the glibc library. It summarizes results with pass or fail for each test for easy comparison, but it also provides a more complete output, which contains the reason for failure.

The test environment used was a Pandaboard development platform, which was selected because of existing support in the shim layer as well as easy to use debugging capabilities. The Pandaboard contains an OMAP4430/4460 System on a Chip (SoC) [11] with a dual core Cortex-A9 CPU, 1GB of DDR2 RAM. The system's kernel is the latest version of VMXL4 and on top of it, a single instance of VMXLinux with Busybox.

LTP was used to evaluate the transition from `arch/l4` to `arch/arm` as well as the transition from 3.0.8 `arch/arm` to 3.4.0 `arch/arm`. The goal was to obtain the same results in the first case and similar if not the same results in the second case. Both these goals were achieved. To be noted, the 3.4.0 native kernel does worse on the LTP test that the paravirtualized one. One of the reasons for this is the fact that the same LTP version was used for both 3.0.8 and 3.4.0, and a number of syscalls were retired between these versions, but were not removed from the paravirtualized Linux. The results are not compared to native 3.0.8 as this was discussed in more detail previously [6] and is not the focus of this article. A summary of the results are found in Table 1: N means native, P means paravirtualized. One of the tests blocked on native 3.4.0 arm and was removed in this case. One of the tests was failing non-deterministically on the 3.0.8 kernel, so it was marked as failure on both (the LTP version has some broken tests [6]).

Table 1

LTP Results

Kernel	3.0.8 l4 P	3.0.8 arm P	3.4.0 arm P	3.4.0 arm N
Failed/Total	71/824	71/824	73/824	82/823

Another factor is the possible regression of performance. LMbench [12] was used. LMbench covers most of the important performance metrics, including bandwidth for pipes, network and file systems and latency for signals, process creation and context switching.

Table 2

LMBench Results

Kernel	3.0.8 l4 P (1)	3.0.8 arm P (2)	(1) / (2)
Fork + Exit (μs)	2666.79	3001.39	0.88
Fork + Execve (μs)	7205.55	7894.75	0.91
Fork + /bin/sh -c (μs)	25930.26	27844.72	0.93
Pagefaults on file write (μs)	39.85	49.11	0.81
AF UNIX sock strem BW (*MB/s*)	89.40	94.33	0.94
Pipe BW (*MB/s*)	51.06	53.87	0.94
TCP/IP connect to local (μs)	486.26	527.99	0.92

LMbench results were very similar. There are some differences in a few areas, found in Tables 2, 3, 4, however, in most tests the differences are not noticeable. The results that differ in a significant way (for example time to fork) can be attributed to different configuration of the Linux kernel, and most likely kernel preemption, which was disabled on `arch/l4` but enabled on `arch/arm`. Preemption lowers throughput, but reduces latency.

Table 3

File System Latency – arch/l4

Size	Created	Creations/s	Removals/s
0k	10750	10828	17495
1k	5774	5266	10147
4k	3952	3790	10404
10k	2419	2172	460

Table 4

File System Latency – arch/arm

Size	Created	Creations/s	Removals/s
0k	7795	6449	14587
1k	4652	4740	7239
4k	3845	3709	7942
10k	2207	2007	228

Since the goal of the project was to reduce the time required for porting and upgrading the Linux kernel, a very important metric was time for porting the new shim layer to a new version of the Linux kernel. This process was discussed in Section 4 and it revealed that the new shim layer is easily upgradeable and might be even easier if instead of jumping versions it would be carried through each version. The results here are very promising, since it took a number of 8-10 person days to get the shim layer working on a new version of Linux, with similar LTP results as the previous version. It is to be noted however, that the same LTP

version was used and the newer system calls were not tested and it is possible that the version of LTP used might have broken or have incomplete coverage.

6. Conclusions

The goal of this project was to improve the portability of the shim layer. The first results are promising and there are a series of modifications which are likely to improve portability further, by removing or reducing the areas where differences occur. Since the new shim layer is easily added to new Linux versions, catching up with the mainline Linux revisions should be easy, if this becomes a desirable feature in the future. Since the shim layer easily integrates with the configuration system, it is possible to easily build the paravirtualized Linux for multiple devices or multiple configurations of the same device, a step up from the old model where different devices would have different source trees.

A newer version of LTP and BusyBox were compiled for the 3.4.0 Linux. LTP revealed a series of issues with the shim layer. So far none of them have been related to the porting process, but were bugs which existed in `arch/l4` which were not covered by the LTP. This is an ongoing effort. This is important for bringing the shim layer up to date in terms of overall functionality.

Another worthwhile effort would be refactoring some of the existent codebase to remove unnecessary functions, variables and declarations. Identifying dead code or replicated code will reduce the size of the patches added, which in turn reduces porting overhead.

Because the build system is nearly identical with the one in native Linux, warnings can also be turned on easily. The L4 architecture did not have this option, which made a series of warnings to accumulate over time. While warnings are not always bugs, they can lead to such. Cleaning up the code could remove a series of hard to detect bugs or reduce the chance for such bugs to appear in the future.

Acknowledgements

The work has been funded by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Ministry of European Funds through the Financial Agreement POSDRU/159/1.5/S/134398.

REFERENCES

[1]. *Cristina Soviany*, Embedding Data and Task Parallelism in Image Processing Applications, PhD Thesis, Technische Universiteit Delft, 2003

[1]. *M. Ciocea*. Paravirtualization of Android Linux on top of VMXL4 Microkernel. Diploma thesis, University POLITEHNICA of Bucharest, July 2012.

[2]. *C. Dall and J. Nieh*. Kvm for arm. In Proceedings of the Ottawa Linux Symposium, Ottawa, Canada, 2010.

[3]. *J.-Y. Hwang, S.-B. Suh, S.-K. Heo, C.-J. Park, J.-M. Ryu, S.-Y. Park, and C.-R. Kim*. Xen on arm: System virtualization using xen hypervisor for arm-based secure mobile phones. In Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE, pages 257–261. IEEE, 2008.

[4]. *M. Lange, S. Liebergeld, A. Lackorzynski, A. Warg, and M. Peter*. L4android: a generic operating system framework for secure smartphones. In Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices, pages 39–50. ACM, 2011.

[5]. *B. Leslie, C. van Schaik, and G. Heiser*. Wombat: A portable user-mode linux for embedded systems. In Proceedings of the 6th Linux. Conf. Au, Canberra, 2005.

[6]. *L. Mogosanu*. Evaluating Virtualization on Top of the VMXL4 Microkernel. Master thesis, University POLITEHNICA of Bucharest, July 2013.

[7]. *G. J. Popek and R. P. Goldberg*. Formal requirements for virtualizable third generation architectures. Commun. ACM, 17(7):412–421, July 1974.

[8]. *L. Rosculete*. Paravirtualizing Android Linux on top of Pandaboard. Diploma thesis, University POLITEHNICA of Bucharest, September 2012.

[9]. *A. Sendroiu*. Hardware Support in Embedded Operating Systems. Master thesis, University POLITEHNICA of Bucharest, July 2013.

[10]. Linux Text Project (LTP). https://github.com/linux-test-project/ltp/wiki [Online; accessed 25 June 2012]

[11]. OMAP4430 SoC. http://www.ti.com/product/omap4430 [Online; accessed 15 May 2013]

[12]. LMbench - Tools for Performance Analysis. http://www.bitmover.com/lmbench/ [Online; accessed 15 May 2013]