CS411 Final: Comparing Android and Linux Kernels

Ryley Herrington December 2, 2012

1 Introduction

The purpose of this final was to point out that there are real world applications to what we have been learning in 411 this term. I will be comparing the Android_x86 (here on out will be referred to as the Android kernel) to the regular Linux kernel we have been modifying all term.

The two main concepts I will be comparing and contrasting will be surrounding the scheduling policies and algorithms and the usb drivers.

2 Scheduling

The scheduling in either kernel is designed to decide which processes will execute next, for how long, with what resources, and it's supposed to do all of this as quickly as possible. The main differences will be regarding application process space because the two systems have different interfaces and overall different capabilities (even if the gap is closing on those).

2.1 Similarities

They both have multiple scheduling algorithms based off of the scheduling that Linux produced starting with v2.6; SCHED_NORMAL and real time scheduling policies are in both kernels. The implementation is nearly identical in both. They both will default to using SCHEDU_NORMAL and have FIFO and RR scheduling that you can specify to use if you need them. And even the FIFO will not use a timeslice like in Linux.

So there are a lot of similarities in the kernels because it's basically the same exact code as in the regular Linux kernel. But similarities are boring. Let's get to the differences.

2.2 Differences

There are some differences in the kernels that are actually interesting even though they seem minor. Often the Linux kernel will give higher priorities to the user interface portion of the systerm because people want a fast response time. The same goes for Android except moreso because mobile phones and tablets (touchscreen devices in general) need to respond quickly or else people will get irritated and throw expensive devices around. You have to remember that a Linux computer has a mouse and usually more CPU and other resources to throw around and Android has to worry about it more.

In Linux we use the scheduler set to Normal most of the time, and will put services that are preempted in the background and work on them the next timeslice they have. The way that Android gets around not having enough resources though is that they have foreground and background service threads. You heard that Android doesnt use thread priorities to reduce how much background work interrupts the user interface, but that's completely wrong. They actually use the same thread priorities as Linux does (even the same scale). In Linux there are background threads running on multiple cores and the system can make forward progress in the background but in android they are given less priority.

User interface threads normally run at the default priority; background threads run in the background priority; not to be circular that's just how they're defined. Application processes that are in the background have all of their threads forced to the background priority.

Androids background priority uses a Linux facility called cgroups to put all background threads into a special scheduling group which, all together, cant use more than 10% of the CPU. For example,

if you have 10 processes in the background all trying to run at the same time, when combined they can't take away more than 10% of the time needed by foreground threads. This is enough to let the background thread make some progress (downloading some music or a video while texting) without having enough of an impact on the foreground threads to be noticed by the user.

Another difference is that Linux doens't run most of it's user processes in a virtual machine. Android has to run Java on their kernel in the Dalvik (J)VM. An Android system will have a set of Unix/Linux processes running constantly. Some are native processes but many will be processes that run a Java virtual machine. These processes usually will be multi threaded and all android threads are native pthreads, so that makes it easier to run in the first place. For example, Calling Thread.setPriority that is part of the standard Java API and contains a value from MIN_PRIORITY(1) to MAX_PRIORITY(10).As all threads are pthreads these priorities will be mapped to unix process priorities (MIN_PRIORITY being 19 and MAX_PRIORITY -8). So there is an interface to make them easier to handle in the JVM, but it's another difference in the scheduling policy even though they seem to be very similar.

3 Drivers

Drivers are a necessary part of any operating system; and the kernel is the interface between the hardware and software side. In Linux we implemented a USB driver that would write to the file descriptor that would point to the USB object and that's that. We could, however, mmap it to a sector in memory and keep that in a DMA zone and write to that and have to USB object read it every so often. We do end up polling, but we end up polling a file and not just a shared memory sector.

3.1 Similarities

If you look in the Android kernel and the Linux kernel you'll notice a file named usb-skeleton.c. This is similar to the missile launcher that we wrote for the final project because they both implement the basic functions you would need for a char device and not a shared memory device. They would both work the same except that the way to access them would be different.

They both will have to have all of the system calls that the usb device would haveto implement and use. In the usb driver there will always have to be a read and write method; even if we don't use it. For example in our project we didn't have to use the read method but we did have to implement a method stub basically.

But Linux and Android use the same protocols to enforce the methods to be present. They will always have a way to ensure that they are present. But, some USB drivers will just allocate some memory and put some shared permissions on the memory so that both can access it (different than what we did with a char device, this would be for a block device).

3.2 Differences

In mm/ashmem.c, you'll see a file that isn't in the Linux kernel. That is because the Android version of shared memory is different than the Linux version. Again the reason for this reimplementation is because most android devices have more limited resources than the computers that usually run Linux.

The ashmem subsystem is a new shared memory allocator, similar to POSIX shared memory but with different behavior regarding how much memory is used. And it has a file-based interface that is simpler to understand than the standard Linux version. It better-supports low memory devices, because it can discard shared memory units under memory pressure from other higher priority tasks.

In the skeleton usb version there are also a few other things that the binder IPC would have to use to bind it to the program that would talk to it through the kernel. Fairly simple things, but it

can pass fd's to the process a little easier than in the kernel because the layer between the kernel and the user space is more locked down in Android. I assume this is because they assume that people will screw up their phones, which is a fair assumption.

4 Group Evaluation

I'll start off and say that Steven Reid and I worked quite well together, and did most of the work on many projects. Steven pointed out to me that we work very well together but about half way through the term and we would get rolling on an idea and implement it by ourselves and kind of exclude William Leslie. It may be unfair to point out that we did most of the work because of that actually. But once Steven pointed that out we tried fixing the issue, so we tried working with Will apart from each other, then we would work together with just us. It seemed to work; but I just feel like we still did most of it. Granted on the final project (the missile launcher) Will was as excited as we were and helped much more.

If anything I would say I was pretty outspoken and that might've scared off Will from speaking up. That could easily be the case; I'm just not sure. Anyway, not trying to skewer any of my team mates, we all helped in our own ways. I'll try to be more accepting of people's ideas and welcoming to shy personalities in the future.