Deploying Preemptible Linux in the Latest Camcorder

Geunsik Lim
Samsung Electronics
geunsik.lim@samsung.com

Jupyung Lee
Samsung Electronics
jupyung.lee@samsung.com

Sangbum Suh
Samsung Electronics
sbuk.suh@samsung.com

Abstract

Currently, the Linux kernel is well equipped to compete with the soft realtime operating system. Linux has been the choices of the operating system. We adjusted optimized Linux kernel to the camcorder's system architecture which is equipped with ARM cortex-A8 and implemented open-source based tool-chain, audio zoom calculation, and realtime HDMI I2C communication and userspace realtime thread program. Samsung has introduced Consumer Electronics Show(CES) this year with its new S-Series of full HD digital camcorders. These product is the world's first commercially available camcorder which includes built-in Wi-Fi and DLNA connectivity.

This paper describes our trouble shooting, cross-compiler issues, technical experiences and best practice in reducing latency in Linux and applications for developing an embedded product like camcorder. This discussion focuses on how commercial platform can optimize the realtime extensions available in Linux kernel, but it is also relevant to any software developer who may be concerned with finding a suitable tradeoff between throughput and responsiveness for embedded systems. Furthermore, many methods which implemented to further improve the system performance will be presented as well.

1 Introduction

Since 2000, commercial RTOSs such as pSOS, LynxOS, QNX, Nucleus, Vxworks that have been used in embedded devices have been replaced by Linux kernel. For home appliances such as Digital TV, camcorder, digital camera, printer, etc., electronic companies have improved their differentiation and competitiveness by modifying open-source software for each product use since they adopted Linux which has the advantage not to require paying royalty. For mobile platform such as mobile phone, variable commercialization projects based on Linux such as Maemo, Moblin, Android, and Palm-Pre are in progress.

Particularly, when starting with Linux version 2.6, realtime functionalities[19] such as O(1) Scheduler, Threaded Interrupt Handling, Preemptible Kernel[8], Priority Inheritance, High Resolution Timer[5], Tickless Timer and Userspace Realtime Mutex[2] have been added so that they are being used for embedded devices which require responsiveness as important factor.

For example, Commercial RHEL-RT[18] that its realtime property is improved by Ingo Molnar[9] who is one of the realtime system developers, currently working for Red Hat has been applied for USS Zumwalt (DDG-1000, U.S. Navy Next-Generation Destroyer) to ensure that computers response correctly without halt & stop of computers or failure of synchronization with other events.

However, when originally development of the Linux kernel began, it was not designed for the purpose of realtime property. That is why Linux may be suitable for the system that required the soft realtime property, not the hard realtime property.

Although Linux is continuously being developed to be closer to the hard realtime property by many open-source developers[1, 23] non-preemptible critical sections and interrupt off sections still exist in the RT patched Linux kernel.

In the case of full HD camcorder, commercial camcorders have been developed by adopting RTOSs such as Vxworks, TronOS instead of Linux for the OS in order to meet realtime property. However, commercialization was attempted for the full HD camcorders from the Samsung camcorder S-Series based on the Linux 2.6.29 kernel considering the advantages that the payment for royalty is not required and the source can be freely modified.

The Samsung camcorder S-Series offer both built-in Wi-Fi and DLNA connectivity and feature Samsung AllShare that allows users to easily access, manage, and share content including their full-HD videos across other DLNA certified devices. The Samsung full HD camcorder is commercially available in the worldwide market.

2 Responsiveness VS. Throughput

The goal of Real Time Operating System(RTOS) is to provide a predictable and deterministic environment. The primary purpose is not to increase the speed of the system[6], or lower the latency between an action and response, although both of these increase the quality of a RTOS[14].

Many companies have basically adopted RT patched Linux kernel for their embedded products recently in order to meet realtime property. When running software in such a system, tradeoff between responsiveness and throughput is often found. Otherwise, responsiveness and throughput are inversely related.

The overhead for realtime preemption is applicable for these cases [16].

- Mutex operations more complex than spinlock operations
- Priority inheritance on mutex increases task switching
- Priority inheritance increases Worst-Case Execution Time(WCET)

And, flexible design allows much better worst case scenarios in embedded products.

- Realtime tasks are designed to use kernel resources in managed ways then delays can be eliminated or reduced

For example, For recording mass files, if we try to unmap memory that we allocated with 100MB for recording in camcorder, we have to wait for more than 3seconds to change mode from play mode to recording mode. This results from the unit of memory unmapped size when we are recording mass files like camcorder particularly.

When performing memory unmap with system call to virtual memory area used by user, it improves realtime property by reducing the size of non-preemptible sections associated with the unmap operation through the minimization of "ZAP_BLOCK_SIZE". However, the memory unmap operation results in delay due to the minimized "ZAP_BLOCK_SIZE". In this case, we recommend that default memory range to unmap allocated memory from 8 pages to 1,024 pages. Figure 1 shows operation comparison between the memory unmap ranges. We can not find side-effect during the sotware quality assurance test after adjusting this approach .

The unmap operations were repeated 3,328 times with 8 pages units in order to reset the 100MB of memory, and check out whether any task with higher realtime priority is waiting every time unmapping is completed in each 8 pages units for realtime property. If any task with higher realtime priority exists, the task preempts the current task and the memory unmap operation will be performed again after finishing the task. At the moment, It takes more than 3 seconds to unmap 100MB of memory with 8 pages units in our test, and the average share of CPU during the time accounted for 92% - 98%.

For a camcorder in this paper, as mass files have to be recorded, memory has to be allocated with 100MB unit and reset. When changing the boundary between responsiveness and throughput to 1,024 pages from 8 pages through repeated tests, it showed the most realistic performance. After changing the policy of memory unmap which changes minimum unit for memory unmapping to 1,024 pages, the switching time from play mode to recording mode in mass production has been improved from 3 seconds and more to less than 1 second. In our experiment, We decided that the best page size of unmap operation was 1,024 pages without the more page size like 2,048 pages and 4,096 pages to consider both throughput and responsiveness.

When unmapping 100MB that was allocated for recording video files through the Software Quality Assurance(SQA) test process before and after changing the unit in camcorders, it showed that the performance of unmap_page_range() with 1,024 page unit is effective to solve the memory unmap operation issue is delayed due to the minimized "ZAP_BLOCK_SIZE".

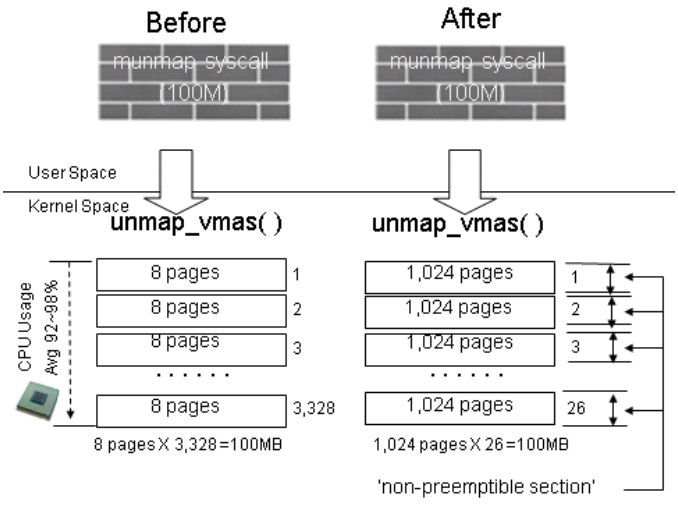

Figure 1: Operation comparison between the memory unmap ranges

3 Cross Compiler to solve low clock speed

While we prepared for the mass production of the Samsung camcorder, we tried that the clock speed for the product should be changed from 800Mhz to 600 Mhz to reduce overall power consumption of camcorder. It means that the lower clock speed is suitable for commercial product if we can. After the change of the clock speed is adjusted, Video recording test is failed on SQA test before production release because of low clock speed.

3.1 S5PC110 hardware specification

Before explaining the problem, We first presents hardware characteristics of Samsung camcorder. The S5PC110 is targeted for small form-factor connected devices such as multimedia intensive smart phones in Figure 2, while the S5PV210 is aimed at portable computing devices such as netbooks that demand high performance and design flexibility.[21]

High speed 3D graphics rendering and high resolution video support are two key differentiating features for advanced mobile devices. Both the S5PC110 and S5PV210 are equipped with a powerful built-in POWERVR SGX 3D graphics engine, licensed from Imagination Technologies, to support sophisticated 3D UI and high-caliber games. In addition, the two processors integrate a1080p full HD codec engine that supports 30fps full HD video playback and recording. A built-in HDMI1.3 interface allows output of captured or downloaded mobile multimedia contents to an external high definition digital display.

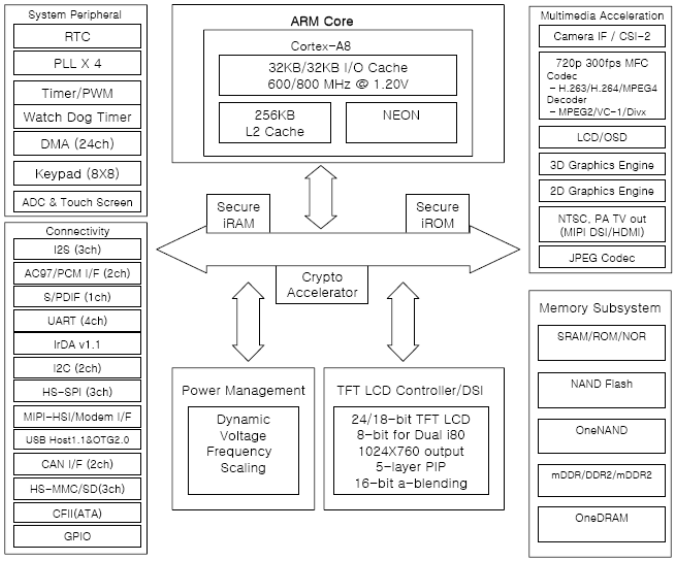

Figure 2: S5PC110(Cortex-A8) Architecture map

3.2 How to overcome low clock speed

First of all, We considered software technologies to minimize code size and optimized throughput according to the compiler option tuning, As a result, we passed 68% with GCC 4.4.3 based tuned toolchain and passed others with updated AudioZoom engine library. Fortunately, we overcomed the problem of low clock speed with software approaches to observe the development schedule that we had to work successfully for commericial product.

3.3 GCC optimization for performance & code size

This optimization can provide significant increases in performance and equally significant reductions in code size for camcorder[4].

The default inline options for camcorder are following.

```
-finline-functions
-fno-inline-functions-called-once
```

Below are options that we found via mobile software platform study like Android[22], Moblin, Maemo. Especially, Parameters setting is a key driver in performance and size optimization. We searched for a configuration that reduces size the most using compiler option search approach.

```
-finline
-fno-inline-functions
-finline-functions-called-once
--param max-inline-insns-auto=50
--param inline-unit-growth=0
--param large-unit-insns=0
--param inline-call-cost=4
```

- -fno-inline: Don't pay attention to the inline keyword. Normally this option is used to keep the compiler from expanding any functions inline. Note that if you are not optimizing, no functions can be expanded inline.

- -finline-functions: Integrate all simple functions into their callers. The compiler heuristically decides which functions are simple enough to be worth integrating in this way. Enabled at level '-O3'.

- -finline-functions-called-once: Consider all static functions called once for inlining into their caller even if they are not marked inline. Enabled at levels '-O1', '-O2', '-O3' and '-Os'.

- max-inline-insns-auto: When you use '-finline-functions' (included in '-O3'), a lot of functions that would otherwise not be considered for inlining by the compiler will be investigated. To those functions, a different limit compared to functions declared inline can be applied. The default value is 50.

- inline-unit-growth: Specifies maximal overall growth of the compilation unit caused by inlining. The default value is 30 which limits unit growth to 1.3 times the original size.

- large-unit-insns: The limit specifying large translation unit. Growth caused by inlining of units larger than this limit is limited by '–param inline-unit-growth'.

- inline-call-cost: Specify cost of call instruction relative to simple arithmetics operations(having cost of 1). Increasing this cost disqualifies inlining of non-leaf functions and at the same time increases size of leaf function that is believed to reduce function size by being inlined. The default value is 12.

3.4 Code size comparison among the compilers

Figure 3 is the comparison as to how the size of userspace library of camcorder changes by each GCC version. When compiling the source of system library by using tuned GCC 4.4.3 version, 6.60% of size reduction was obtained compared to existing GCC 4.2.0 version in Figure 3.

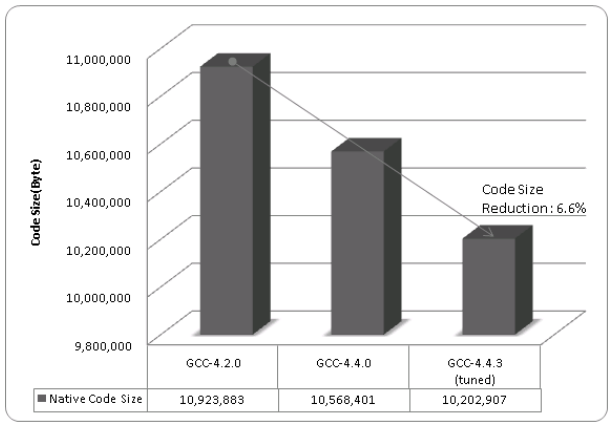

Figure 3: Code size comparison among the cross compilers

3.5 Arithmetic speed for recording functions

Below is evaluation result among the cross compilers to solve the low clock speed problem when running recording function. From the test result, the arithmetic speed of AudioZoom library engine for recoding functions by using tuned GCC 4.4.3 version was increased by 10% compared to existing performance speed in Table 1.

- Test environment: Preemptible Linux + Lightweight root file system for embedded device
- GCC 4.2.X option: -march=armv7 -O3 -ffast-math -fomit-frame-pointer -funroll-all-loops -pipe
- GCC 4.4.3 option: -finline -fno-inline-functions -finline-functions-called-once –param max-inline-insns-auto=50 –param inline-unit-growth=0 –param large-unit-insns=0 –param inline-call-cost=4

3.6 CPU usage comparison

AudioZoom library performs arithmetic functions and the largest computing tasks when recording video on the camcorder. Figure 4 shows CPU usage monitoring result after moving latest GCC version and optimization.

GCC Version	*CodeSourcery(Lite) 4.2.1*	*CrossTool-0.43 4.2.0*	*Montavista 5.0 4.2.0*	*Opensource(tuned) 4.4.3*
Binary Size	175,034	175,034	170,377	156,019
1st Test	12,970,182	13,138,915	13,258,427	12,060,383
2nd Test	13,030,439	13,163,295	13,212,784	12,157,852
3rd Test	13,345,300	13,038,169	13,075,400	12,279,066
4th Test	12,889,146	13,353,949	13,171,671	12,277,243
5th Test	13,000,646	13,147,129	13,158,650	12,192,311
Average	13,047,143	13,168,291	13,175,386	12,193,371
Time(Sec)	1	1,009285	1.001402	0.916656

Table 1: Performance result of AudioZoom engine among the cross compilers

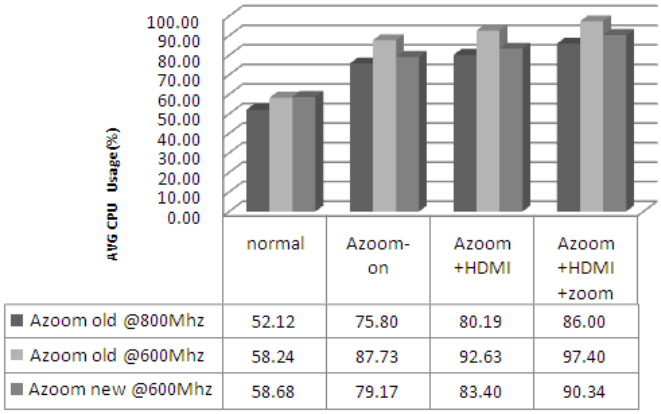

Figure 4: CPU usage comparison after improvement

3.7 SQA Test after changing cross compiler

After changing GCC version to 4.4.3, optimizing the inline function, and improving "Azoom(AudioZoom)" library engine for arithmetic process in recording, video recording for 16 SD cards from different manufacturers was normally performed in Software Quality Assurance(SQA) test though the clock speed of CPU was lowered from @800MHz to @600MHz like Table 2.

3.8 GPL license issues

As we all know, Glibc open-source community selected to encourage open-source based commercial products. Our GCC 4.4.3 based tuned toolchain consists of Glibc 2.11. Recently, Glibc 2.6.1+ is "LGPLv2+ and LGPLv2+ with exceptions and GPLv2+". Many companies want to close sources of userspace application of camcorder for competitiveness thereby selecting LGPL based Glibc open-source software. We can release our software with "GCC 4.4.3 + Glibc 2.11" because some sources of glibc are that defined with GPLV2+ are as Executable Linking File(ELF) binary for debugging and configuration setting only without shared object('x.so').

When compiling the Glibc source files that GPLv2 is applied in Glibc 2.11 source, 'x.so' files for library use in rootFS(root file system) don't exist. However, as they are ELF Binary files for debugging and test, license issue for application codes open doesn't come out when using glibc 2.6.1 version or higher to embedded product. The source that requires GPLV2 license applied for shared library('x.so') files has not been used so far, as seen in Table 3. If GPLV2 is adopted to some shared library('x.so') files of the specified version that was updated for new feature and bugfix in the future, another C library like uClibc, eGlibc, BSD C library should be considered in order to protect the application codes developed among the companies for commercial products from being disclosed.

4 Needs to move the latest Linux version

We use the term backporting to describe the situation case where we take a fix for a security flaw out of the most recent version of an upstream software package, and apply that fix to an older version of the package we release for server solutions and embedded devices like RHEL(Redhat Enterprise Linux), Montavista [26], Windlinux.

Backporting is the action of taking a certain software modification (patch) and applying it to an older version of the software than it was initially created for. It is part of the maintenance step in a software development

Maker	*Type*	*Cap*	*Write(Kb/s)*	*800Mhz*	*600Mhz*	*600Mhz*	*600Mhz*
				AudioZoom Ver 1.18	AudioZoom Ver 1.18	AudioZoom Ver 1.18	AudioZoom Ver 1.22
				GCC 4.2.0 (Montavista)	GCC 4.2.0 (Montavista)	GCC 4.4.3 (tuned)	GCC 4.4.3 (tuned)
A-DATA	SDHC	8GB	10,556	ok	fail	ok	ok
AnyFlash	SDHC	8GB	8,687	ok	fail	ok	ok
imation	SDHC	8GB	9,945	ok	fail	ok	ok
imation	microSD	8GB	5,981	ok	fail	ok	ok
inx	SDHC	8GB	5,876	ok	fail	ok	ok
Jaydisk	SDHC	8GB	8,802	ok	fail	ok	ok
Memorette	SDHC	8GB	10,343	ok	fail	fail	ok
Memorette	SDHC	8GB	7,062	ok	fail	ok	ok
Memory4U	SDHC	8GB	8,260	ok	fail	fail	ok
Samsung	SDHC	8GB	9,690	ok	fail	ok	ok
Sandisk	SDHC	8GB	8,148	ok	fail	fail	ok
Sandisk	SDHC	8GB	11,725	ok	fail	ok	ok
Sandisk	SDHC	8GB	8,148	ok	fail	fail	ok
Timu	SDHC	8GB	10,172	ok	fail	ok	ok
Toshiba	SDHC	8GB	5,873	ok	fail	fail	ok
Toshiba	SDHC	8GB	10,072	ok	fail	ok	ok

Table 2: Recording test among the 16 SD cards

process. It's important that we examine how the Linux kernel source changed recently.[27]

When considering the cost for labor and total costs for the period of development in regard to R&D for commercial embedded products, change of existing released kernel version that has been commercialized and well working into the latest version can be a risk to companies in terms of stability. For this reason, many embedded product manufacturers are hesitating to change the existing Linux kernel version to the latest version.

However, if an existing Linux kernel version is applied for new products, it has a drawback that debugged functions and new kernel features can't be used. If the fact that new Linux kernel version is actually released every couple of months is taken into account, backporting can be a practical choice as a method for considering both product stability and new kernel features.

4.1 Case study: BUG: swapper: xxxxxx Error message when booting

- Problem: "BUG:swapper: 1 task might have lost a preemption check!" message.
- Reason: When we enabled 'CONFIG_DEBUG_PREEMPT' feature, Kernel display the status of preemption mode every times When booting.
- Solution: Backporting from Linux 2.6.30-RT

4.2 Case study: Allocation error with GCC 4.4.X

- Problem: Relocation error when we compile linux-2.6.29 Sources with GCC 4.4.3 toolchain (Undefined relocation type in Linux kernel module *module.c*)

  Relocation section '.rel.text' at offset 0x1adae0 contains 1960 entries:

```
Offset          Type           Sym. Name
00000088   R_ARM_MOVW_ABS_NC  jiffies
00000090   R_ARM_MOVT_ABS     jiffies
000001f4   R_ARM_MOVW_ABS_NC  .LANCHOR
000000200  R_ARM_MOVT_ABS     .LANCHOR
000000264  R_ARM_ABS32        .text
```

- Reason: Linux 2.6.29 don't define relocation type in module.c for GCC 4.4.X
- Solution: Backporting from Linux 2.6.30

Source files under GPLV2+	*Binary file name*	*Description*
elf/ldconfig.c, elf/readlib.c	ldconfig	ELF file
elf/cache.c, elf/chroot_canon.c	ldconfig	ELF file
nscd/*.c (22files)	nscd	A Name Service Caching daemon
catgets/gencat.c	getcat	ELF file to create formatted msg catalog
locale/programs/*.c (38 files)	locale, localedef libBrokenLocale.so	iconv config file
posix/getconf.c	getconf	Binary to get glibc setting information
iconv/dummy-repertoire.c	iconv_prog, iconvconfig	Iconv ELF file, Iconv config file
iconv/iconv_charmap.c	iconv_prog, iconvconfig	Iconv ELF file, Iconv config file
iconv/iconvconfig.c	iconv_prog, iconvconfig	Iconv ELF file, Iconv config file
iconv/iconv_prog.c	iconv_prog, iconvconfig	Iconv ELF file, Iconv config file
malloc/memusagestat.c	memusagestat	ELF file
sysdeps/.../linux/nscd_setup_thread.c	nscd	To create Thread on nscd Daemon

Table 3: Glibc 2.11 source files under GPLV2+

4.3 Case study: Undefined reference to '_gnu_mcount_nc'

- Problem: We often enable 'ftrace(function tracer)' feature for tracing internal behavior of Linux. But We can't compile Linux 2.6.29 with GCC 4.4.X

```
...undefined ref to `__gnu_mcount_nc'
...undefined ref to `__gnu_mcount_nc'
.../mach-c110/built-in.o: In function
...undefined ref to `__gnu_mcount_nc'
...undefined ref to `__gnu_mcount_nc'
...more undefined ref to
`__gnu_mcount_nc' follow
make: *** [.tmp_vmlinux1] Error 1
```

- Reason: Since GCC 4.4's the name and calling convention is changed for function profiling like ftrace on ARM.

- Solution: Backporting from `http://marc.info/?l=linux-arm-kernel&m=124946219616111&w=2` to support new EABI compatible profiler *__gnu_mcount_nc*. With this patch both types are supported. Since the first submission We folded in the EXPORT_SYMBOL changes by Anand and removed the question mark after "gcc 4.4". (Using __gnu_mcount_nc was introduced in r140974 to gcc- Fix ARM EABI profiling.)

4.4 Case study: Compiler error of Linux-2.6.29-RT with SLUB features

- Problem: We can't boot Linux kernel that patched Ingo' RT-Patch with SLUB(Unqueued Allocator) features by default.

- Reason: This reason is summarized by Jonathan Corbet on Oct-03-2007. Quite a few other changes can be found in this tree. The SLUB allocator is not an option for Realtime kernels. Instead, this tree uses a modified version of the slab allocator which replaces interrupt-based single-CPU locking with a set of specific per-CPU locks.

  The global files_lock has been removed in favor of tightly-locked per-CPU lists. To help with the creation of such lists, a new locked-list type has been added. The tasklet code has been reworked for better threading, but the tasklet elimination patch is not present. There's also quite a few architecture-specific patches adding various features (such as clock_events) needed by the Realtime tree and fixing problems.

- Solution: Understanding the Realtime tree about SLUB/SLAB[11]

5 IRQ Handler

Under Linux, hardware interrupts are called IRQ's (Interrupt Requests).[17] There are two types of IRQ's,

short and long. A short IRQ is one which is expected to take a very short period of time, during which the rest of the machine will be blocked and no other interrupts will be handled. A long IRQ is one which can take longer, and during which other interrupts may occur but not interrupts from the same device. If at all possible, it's better to declare an interrupt handler to be long.

When the CPU receives an interrupt, it stops whatever it's doing, saves certain parameters on the stack and calls the interrupt handler.

This means that certain things are not allowed in the interrupt handler itself, because the system is in an unknown state. The solution to this problem is for the interrupt handler to do what needs to be done immediately, usually read something from the hardware or send something to the hardware, and then schedule the handling of the new information at a later time (this is called the "bottom half") and return. The kernel is then guaranteed to call the bottom half as soon as possible – and when it does, everything allowed in kernel modules will be allowed.

How can we run IRQ Processing with preemptible linux on camcorder? The realtime for Linux patchset does not guarantee adequate realtime behavior for all target platforms. When using realtime Linux on a new platform you should expect to have to tune the kernel and drivers to provide performance that matches your specific requirements.[3]

In our case, Camcorders have to finish hard IRQ processing most rapidly after HARD IRQ need Realtime characteristics obtain CPU resources above all. And, an architect designed to work lightweight tasks as hardware about video decoding.

If we consider realtime characteristics about entire system in general, threaded hard interrupt is a good choice. But, we have to proceed Hard Interrupt case most rapidly in our Camcorder. Otherwise, our camcorder has to support realtime responsiveness assurance that don't must delay Hardware Interrupt in Figure 5.

For example, when we connect interface between Digital TV and HDMI(High-Definition Multimedia Interface), communication start between DDC(Display Data Channel) and I2C. If the delay happen when IRQ happen, abnormal data transfer often happen by timeout on Samsung Digital TV.

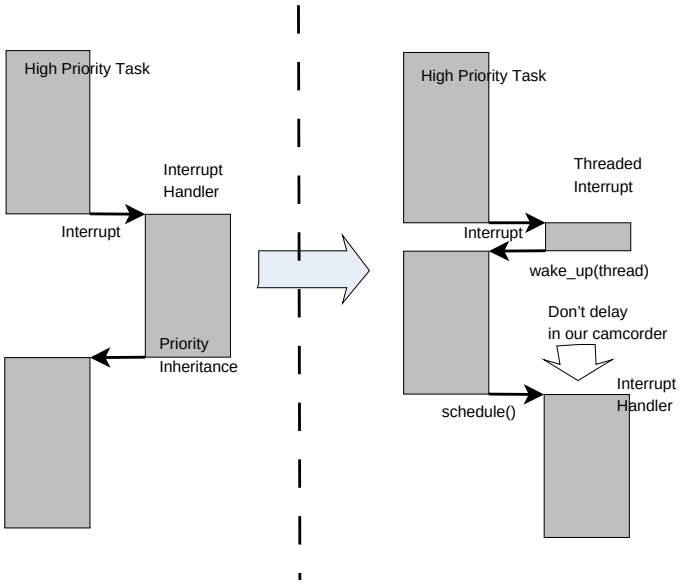

Figure 5: Area for preventing Hard IRQ Delay in Samsung camcorder

"Thread Hardirqs" option reduces the latency of the kernel by "Thread Hardirqs". This means that all hardirqs will run in their own kernel thread context. While this helps latency, this feature can also reduce performance. We don't enable "Thread Hardirqs" in our Camcorder for lightweight Hard IRQ processing at "make menuconfig" menu.[13]

We measured IRQ execution time to decide finally before we disable "Hard thread IRQ" feature in Table 4. For test, we save the execution time with do_gettimeofday( ) in asm_do_IRQ, run `"target> cat /proc/interrupts"` after modifying show_interrupt( ). At the Table 4, 'Deadline(RT-irq)' field means RT-irqs that don't have to be delayed by the higher priority task during the IRQ processing.

6 Further Enhancements

"Complete Preemption Mode" does not mix with 3rd Party's binary kernel modules in our development for commercial full HD camcorder.

We selected "Preemptible Mode" because of 3rd Party's preemptible device driver verification issues. In order to further reduce the impact of human error on driver reliability, we will consider "device drivers verification process" of third party for improved realtime characteristics as further enhancements on next products.

According to the NICTA research, Figure 6 shows a summary of their study of 500 bugs found in Linux device drivers.[12]

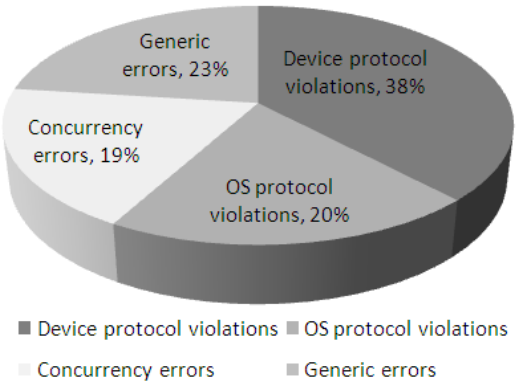

Figure 6: Summary of bugs found in Linux device drivers

We focus on three types of bugs. device protocol violations, i.e when the driver behaves in a way that violates the required hardware protocol, and concurrency bugs, i.e. race conditions and deadlocks resulting from incorrect synchronization of OS threads inside the driver[10], and OS protocol violations, i.e. situations when the driver fails to behave the way the OS expects it to. The common property of these types of bugs is that both of them are related to how the driver interacts with the OS.[20]

Where do bugs come from?

- > 70% of the kernel code is in drivers.
- Drivers contain 3x - 7x bugs per loc compared to the rest of the kernel.

Since device specifications are independent of any operating system, drivers for different systems can be synthesised from a single device specification. As a result, the likelihood of errors due to incorrect specifications will be reduced because these specifications are shared by many drivers[15].

7 Conclusions

In contrast to the origianl Linux for non-realtime environment, embedded developers often need a high level concurrency in their thread application based on real-time priority, with predictable application response to system events rapidly.

Camcorder system requires realtime operating system for deterministic control and rich operating system for running camcorder applications (e.g: wireless, tcp/ip protocol) and exploiting legacy libraries.

Linux kernel subsystem for Camcorder's recording is implemented using memory unmapping approach for realtime control of recording mode and play mode. And, We figiured out needs to move the lates Linux and GCC version according to the experience of backporting from recent version for the enhancement and bug fixes we need is always in the next revision.

Camcorder specific realtime application logic is implemented with many threads in NPTL(Native Posix Thread Library[25]) model based userspace with priority queueing, robust FUTEX[24], priority inheritance[7].

For commercial Camcorder, Linux with proper tuning showed satisfactory performance for a preemptible linux based Camcorder system including RT-patch of Ingo Molnar and others. Application of low latency and preemptible kernel patches make it possible to record video files normally.

References

[1] Real-Time Linux Wiki. Project site: `http://rt.wiki.kernel.org`.

[2] RT-mutex subsystem with PI support. Linux kernel documentation: `kernel/Documentation/rt-mutex.txt`.

[3] Sony Frank Rowand. Adventures in real-time performance tuning. In *Embedded Linux Conference*, 2008.

[4] FSF & GNU. *GCC 4.4.4 online documentation*, April 2010.

[5] Thomas Gleixner and Douglas Niehaus. Hrtimers and beyond: Transforming the linux time subsystems. In *Ottawa Linux Symposium*, 2006.

[6] Darren V. Hart. realtime linux latency comparisons. In *Ottawa Linux Symposium*, 2007.

[7] Ingo Molnar. PI-futex. people.redhat.com: `http://lwn.net/Articles/102216/`.

[8] Ingo Molnar. remove the Big Kernel Lock, this time for real. Linux kernel mailing list: http://lwn.net/Articles/102216/.

[9] Ingo Molnar. RT-patch. Index of /mingo/realtime-preempt:http://www.kernel.org/pub/linux/kernel/projects/rt/.

[10] Jonathan Corbet. Driver porting: completion events. Linux Weekly News: http://lwn.net/Articles/102216/.

[11] Jonathan Corbet. What's in the Realtime tree. Linux Weekly News: http://lwn.net/Articles/102216/.

[12] Ihor Kuz Leonid Ryzhyk, Peter Chubb and Gernot Heiser. Dingo: Taming device drivers. In *Proceedings of the 4th EuroSys Conference, Nuremberg, Germany*, 2009.

[13] Linus Tovalds. The Linux Kernel Archives. kernel.org: http://www.kernel.org.

[14] Paul E McKenny. 'real time' vs 'real fast': How to choose? In *Ottawa Linux Symposium*, 2008.

[15] Microsoft. *Architecture of the kernel-mode driver framework(KMDF),2006/2007*, 2007.

[16] Montavista. *Real-time Application Programmer's Guide*, 2009.

[17] Peter Jay Salzman. The Linux kernel Module Programming Guide. TLDP: http://tldp.org/LDP/lkmpg/2.4/html/.

[18] Redhat. *Red Hat Enterprise MRG 1.2 Realtime Tuning Guide*, 2009.

[19] Steven Rostedt. Internals of the rt patch. In *Ottawa Linux Symposium*, 2007.

[20] Leonid Ryzhyk. On the construction of reliable device drivers. In *PhD Thesis, School of Computer Science and Engineering, University of NSW*, 2010.

[21] SAMSUNG. SAMSUNG Opens the Door to PC-Level Performance on Mobile Devices with 1Ghz Low-power Application Processors. Samsung News: http://www.samsung.com/global/business/semiconductor/newsView.do?news_id=1043.

[22] Shih-wei Liao. Smaller and Faster Android. COSCUP(Conference for Open Source Coders, Users and Promotoers): http://coscup.org/2009/zh-tw/program/.

[23] Thomas Gleixsner. Cyclictest. http://rt.wiki.kernel.org/index.php/Cyclictest.

[24] Ulrich Drepper. Futexes Are Tricky. people.redhat.com: http://people.redhat.com/drepper/futex.pdf.

[25] Ulrich Drepper. The native POSIX Thread Library for Linux. people.redhat.com: http://people.redhat.com/drepper/nptl-design.pdf.

[26] Klaas van Gend. Using realtime linux common pitfalls, tips & tricks. In *Embedded Linux Conference*, 2008.

[27] Wikipedia Contributors. Backporting terminology. Wikipedia, the free encyclopedia: http://en.wikipedia.org/wiki/Backporting.

IRQ No.	*Deadline(RT-irq)*	*Chip Name*	*Action Name*	*Description*
16	-	s3c-uart	s5pc100-uart	UART ch0 RX
18	-	s3c-uart	s5pc100-uart	UART ch0 TX
26	-	s3c-uart	NULL	UART ch2 TX
45	500(Passed)	s3c_vic_eint	hpd	HDMI Hot Plug Detect
50	-	sVIC	PDMA0	Physical DMA Ch-0
51	900(Passed)	VIC	PDMA1	Physical DMA Ch-1
52	-	VIC	samspi-dma	SPI DMA
53	-	s3c-timer	pwm-tick	pwm timer0 for log key
54	-	s3c-timer	pwm-tick	pwm timer1 for time-print
55	-	s3c-timer	pwm_timer2	pwm timer2 for hrt clock_source
57	-	s3c-timer	pwm_timer4	pwm timer4 for hrt tick-timer
73	-	VIC	Pata	ATA
74	-	VIC	3D	3D
76	-	VIC	HDMI	HDMI
79	-	VIC	sam-spi	spi ch 0
80	-	VIC	sam-spi	spi ch 1
81	-	VIC	sam-spi	spi ch 2
83	-	VIC	s3c2410-i2c.2	i2c ch2 for audio-DAc
88	-	VIC	s3c-udc	usb OTG
93	-	VIC	s3c-csis	MIPI
96	-	VIC	s3cfb	LCD[0]
97	500(Passed)	VIC	s3cfb	LCD[1]
101	100(Passed)	VIC	s3c-fimc0	fimc0
102	100(Passed)	VIC	s3c-fimc1	fimc1
103	100(Passed)	VIC	s3c-fimc2	fimc2
104	-	VIC	s3c-jpg	jpeg
106	-	VIC	PowerVR	ad converting
107	100(Passed)	VIC	s5p-tvout	tv mixer
109	100(Passed)	VIC	s5p-ddc	i2c ch1 for HDMI ddc
110	-	VIC	s3c-mfc	mfc
112	-	VIC	s3c-i2s-v50	i2s v50 for audio ch0
113	-	VIC	s3c-i2s-1	i2s for audio ch1
118	-	VIC	s5pc1xx_spdif	spdif
130	500(Passed)	VIC	MMC0	mmc0 for wireless lan
131	-	VIC	s5p-cec	HDMI CEC

Table 4: Deadline measurement result of IRQs that need realtime characteristic after disabling "Hard thread IRQ"

.