AUTOSTATIC.COM

LINUX AUDIO BABBLE

RPI 3 AND THE REAL TIME KERNEL

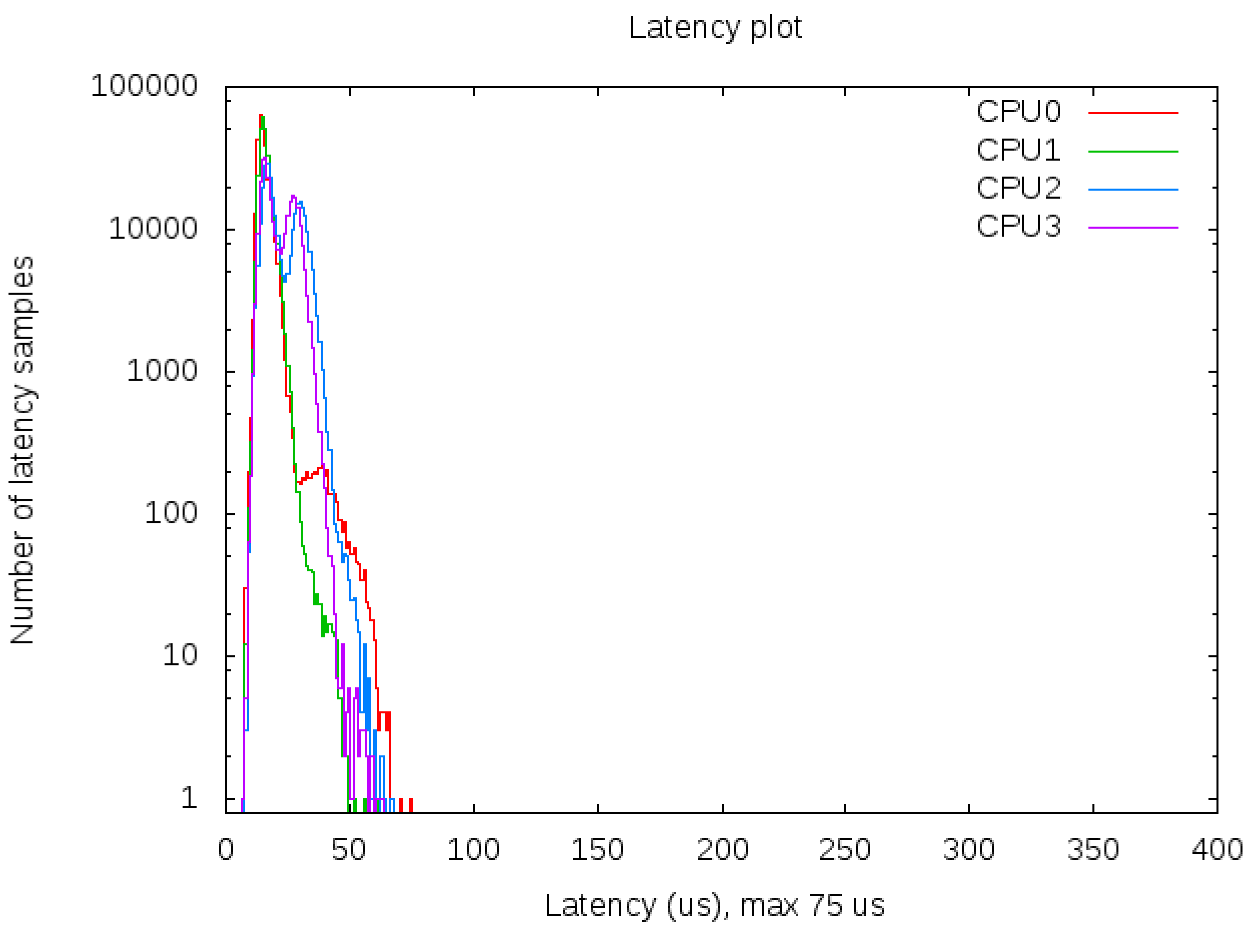
June 27, 2017 | jeremy | audio, cyclictest, hackbench, howto, kernel, linux, linux_audio, manual, midi, mkknlimg, mod, modduo, pi, raspberry, raspberrypi, raspbian, real, realtime, rpi, rpi3, sidecar, time, usb

As a beta tester for MOD I thought it would be cool to play around with netJACK which is supported on the MOD Duo. The MOD Duo can run as a JACK master and you can connect any JACK slave to it as long as it runs a recent version of JACK2. This opens a plethora of possibilities of course. I'm thinking about building a kind of sidecar device to offload some stuff to using netJACK, think of synths like ZynAddSubFX or other CPU greedy plugins like fat1.lv2. But more on that in a later blog post.

So first I need to set up a sidecar device and I sacrificed one of my RPi's for that, an RPi 3. Flashed an SD card with Raspbian Jessie Lite and started to do some research on the status of real time kernels and the Raspberry Pi because I'd like to use a real time kernel to get sub 5ms system latency. I compiled real time kernels for the RPi before but you had to jump through some hoops to get those running so I hoped things would have improved somewhat. Well, that's not the case so after having compiled a first real time kernel the RPi froze as soon as I tried to run`apt-get install rt-tests`. After having applied a patch to fix how the RPi folks implemented the FIQ system the kernel compiled without issues:

```
Linux raspberrypi 4.9.33-rt23-v7+ #2 SMP PREEMPT RT Sun Jun 25 09:45:58 CES
```

And the RPi seems to run stable with acceptable latencies:



Histogram of the latency on the RPi with a real time kernel during 300000 cyclictest loops

So that's a maximum latency of 75 μs, not bad. I also spotted some higher values around 100 but that's still okay for this project. The histogram was created with mklatencyplot.bash. I used a different invocation of `cyclictest` though:

```
cyclictest -Sm -p 80 -n -i 500 -l 300000
```

And I ran `hackbench` in the background to create some load on the RPi:

```
(while true; do hackbench > /dev/null; done) &
```

Compiling a real time kernel for the RPi is still not a trivial thing to do and it doesn't help that the few howto's on the interwebs are mostly copy-paste work, incomplete and contain routines that are unclear or even unnecessary. One thing that struck me too is that the howto's about building kernels for RPi's running Raspbian don't mention the `make deb-pkg` routine to build a real time kernel. This will create deb packages that are just so much easier to transfer and install then rsync'ing the kernel image and modules. Let's break down how I built a real time kernel for the RPi 3.

First you'll need to `git clone` the Raspberry Pi kernel repository:

```
git clone -b 'rpi-4.9.y' --depth 1 https://github.com/raspberrypi/linux.git
```

This will only clone the `rpi-4.9.y` branch into a directory called `linux` without any history so you're not pulling in hundreds of megs of data. You will also need to clone the tools repository which contains the compiler we need to build a kernel for the Raspberry Pi:

```
git clone https://github.com/raspberrypi/tools.git
```

This will end up in the `tools` directory. Next step is setting some environment variables so subsequent `make` commands pick those up:

```
export KERNEL=kernel7
export ARCH=arm
export CROSS_COMPILE=/path/to/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueak
export CONCURRENCY_LEVEL=$(nproc)
```

The `KERNEL` variable is needed to create the initial kernel config. The `ARCH` variable is to indicate which architecture should be used. The `CROSS_COMPILE` variable indicates where the compiler can be found. The `CONCURRENCY_LEVEL` variable is set to the number of cores to speed up certain `make` routines like cleaning up or installing the modules (not the number of jobs, that is done with the `-j` option of `make`).

Now that the environment variables are set we can create the initial kernel config:

```
cd linux
make bcm2709_defconfig
```

This will create a `.config` inside the `linux` directory that holds the initial kernel configuration. Now download the real time patch set and apply it:

```
cd ..
wget https://www.kernel.org/pub/linux/kernel/projects/rt/4.9/patch-4.9.33-r
cd linux
xzcat ../patch-4.9.33-rt23.patch.xz | patch -p1
```

Most howto's now continue with building the kernel but that will result in a kernel that will freeze your RPi because of the FIQ system implementation that causes lock ups of the RPi when using threaded interrupts which is the case with real time kernels. That part needs to be patched so download the patch and dry-run it:

```
cd ..
wget https://www.osadl.org/monitoring/patches/rbs3s/usb-dwc_otg-fix-system-
cd linux
patch -i ../usb-dwc_otg-fix-system-lockup-when-interrupts-are-threaded.patc
```

You will notice one hunk will fail, you will have to add that stanza manually so note which hunk it is for which file and at which line it should be added. Now apply the patch:

```
patch -i ../usb-dwc_otg-fix-system-lockup-when-interrupts-are-threaded.patc
```

And add the failed hunk manually with your favorite editor. With the FIQ patch in place we're almost set for compiling the kernel but before we can move on to that step we need to modify the kernel configuration to enable the real time patch set. I prefer doing that with `make menuconfig`. You will need the `libncurses5-dev` package to run this commando so install that with `apt-get install libncurses5-dev`. Then select `Kernel Features - Preemption Model - Fully Preemptible Kernel (RT)` and select `Exit` twice. If you're asked if you want to save your config then confirm. In the `Kernel features` menu you could also set the the timer frequency to `1000 Hz` if you wish, apparently this could improve USB throughput on the RPi (unconfirmed, needs reference). For real time audio and MIDI this setting is irrelevant nowadays though as almost all audio and MIDI applications use the hr-timer module which has a way higher resolution.

With our configuration saved we can start compiling. Clean up first, then disable some debugging options which could cause some overhead, compile the kernel and finally create ready to install deb packages:

```
make clean
scripts/config --disable DEBUG_INFO
```

```
make -j$(nproc) deb-pkg
```

Sit back, enjoy a cuppa and when building has finished without errors deb packages should be created in the directory above the `linux` one. Copy the deb packages to your RPi and install them on the RPi with `dpkg -i`. Open up `/boot/config.txt` and add the following line to it:

```
kernel=vmlinuz-4.9.33-rt23-v7+
```

Now reboot your RPi and it should boot with the realtime kernel. You can check with `uname -a`:

```
Linux raspberrypi 4.9.33-rt23-v7+ #2 SMP PREEMPT RT Sun Jun 25 09:45:58 CES
```

Since Rasbian uses almost the same kernel source as the one we just built it is not necessary to copy any dtb files. Also running `mkknlimg` is not necessary anymore, the RPi boot process can handle vmlinuz files just fine.

The basis of the sidecar unit is now done. Next up is tweaking the OS and setting up netJACK.

Edit: there's a thread on LinuxMusicians referring to this article which already contains some very useful additional information.

← Moved to Fuga

16 THOUGHTS ON "RPI 3 AND THE REAL TIME KERNEL"

HANS LURCH SAYS: Hi autostatic,

I am very interested in running a low-latency audio realtime System on my RPi 3 as well, but

“Compiling a real time kernel for the RPi is still not a trivial thing to do…”,

so would you maybe share your build?

Btw.:
In your article you point out the approached latency of about 75 µs, but not the corresponding value of the former plain-vanilla Raspbian, so the achieved benefit remains somewhat unclear.

And does the realtime system increase powerconsumption, and thus temperature of the RPi considerably? Does it need enhanced cooling now?

Thank you very much for your answers – and the great work!
Best regards
HL

July 17, 2017 at 1:54 PM • *Reply »*

JEREMY SAYS: Hello Hans, apologies for the belated reply 🙁 I can upload my builds but without any guarantee that it will work for others. Also I won’t provide any security updates or support for the builds.
Good point about vanilla Raspbian test results, I will upload those. And using a realtime kernel normally doesn’t increase power consumption, it’s pinning the CPU to the performance governor and overclocking the RPi that makes the RPi more power hungry. I’ve never needed any extra cooling though.

July 30, 2017 at 11:40 AM • *Reply »*

THIJS SAYS: Hi Jeremy

I like this idea very much!
I did some test a long time ago (everything was still in black & white 😉 with netjack. The idea was exactly the same as you describe : move some CPU hungry plugins to an external server. It worked but then broke by upgrading or something like that.

The MOD however, so far, seems pretty solid so it might be a good time to retry the netjack approach. Combined with a dedicated processing unit this might be a winner.

Good luck and please keep us posed on your progress !

btw, i assume you are using the rj45 connector of the MOD to connect to the RPI?
for some reason i assumed this was some kind of proprietary connection dedicated for the footswitches, but i guess that is incorrect, or is it ?

July 30, 2017 at 1:03 PM • *Reply »*

JEREMY SAYS: Hello Thijs,

I'm using the USB connection, the RJ45 connection only speaks the MOD Control Chain protocol. The Control Chain protocol is an open standard, so not proprietary: http://wiki.moddevices.com/wiki/Control_Chain

October 7, 2017 at 11:50 AM • *Reply »*

JOHN DEY SAYS: How quickly Howto get out of date. I was able to cobble my way through your instructions finding and downloading files that would not download but I didn\'t fully understand the fix-system lockup patch. Things started to fall apart with make menuconfig:
HOSTCC scripts/kconfig/mconf.o
In file included from scripts/kconfig/mconf.c:23:0:
scripts/kconfig/lxdialog/dialog.h:38:20: fatal error: curses.h: No such file or directory
compilation terminated.

It would be great is you could update the Howto. Or, as has been suggested, share your build.

Thanks so very much.

John

October 4, 2017 at 2:48 AM • *Reply »*

STEFAN SAYS: A more recent version of the patch is available here, based on a 4.8 kernel instead of 4.6. It seems to apply fine to the current 4.9 branch.

https://github.com/fedberry/kernel/blob/master/usb-dwc_otg-fix-system-lockup-when-interrupts-are-threaded.patch

October 5, 2017 at 1:27 AM • *Reply »*

JEREMY SAYS: Hello Stefan,

Thanks for the link, I'll check it out and incorporate it in the article.

October 7, 2017 at 12:01 PM • *Reply »*

STEFAN SAYS: Oh and curses.h is the header of libncurses, so you need to install that first (-dev).

October 5, 2017 at 1:29 AM • *Reply »*

JEREMY SAYS: Hello John,

Like Stefan said you need to install `libncurses5-dev` to run `make menuconfig`. I've added it to the article. Thanks both of you guys for pointing that out!

October 7, 2017 at 11:55 AM • *Reply »*

JED BEANS SAYS: Thank you Jeremy for your inspirational exploration into rt raspberry pi music. It's frustrating to me that even thought Jackd2 package is installed with the latest raspbian distro it fails to run straight out of the gate making awesome projects like the Collidiscope along with it's instructions, inaccessible to the non-programmers like myself. I would like to try your instructions, but as other comments have suggested (and my past failures trying these out), I would much prefer to copy a pre-made img of raspbian my micro-sd with rt already pre-configured for my rasberry pi 3.

Is that a possibilty yet? Or should I try soldiering through these above instructions? I have had no luck so far with the other getting jack to run on rasberry pi instructions on linux muscian and other places on the web. My hunch is that the newer distributions of rasbarian screw up the old methods of getting it to work. Anyway thank you very much again!

October 21, 2017 at 8:05 PM • *Reply »*

JED BEANS SAYS: Also, is this solution which includes a precompiled kernel not current anymore? http://www.frank-durr.de/?tag=raspberry-pi-2

October 21, 2017 at 8:18 PM • *Reply »*

JED BEANS SAYS: Well, I gave it a shot and sure enough the first command failed with this message:

```
"error: RPC failed; curl 56 GnuTLS recv error (-54): Error in the pull function.
fatal: The remote end hung up unexpectedly
fatal: early EOF
fatal: index-pack failed"
```

Any suggestions?

October 24, 2017 at 8:02 PM • *Reply »*

OLEG SAYS: Hi,

i have got two questions to your kernel upgrade instructions:
1. How to add failed hunk manually of dry-run patch? Add the particular code of patch to the failed file (considering the line)? In my case this particular code was already inserted but the failed message appeared.
2. What do you mean by \” Copy the deb packages to your RPi and install them on the RPi with dpkg -i\”?
Please help me. Really need RT Kernel.

Thank you!

January 22, 2018 at 11:00 PM • Reply »

ZORGOZ SAYS: I have a suggestion to your first question as I was looking at the source and found no reason why it should fail there. Just add –ignore-whitespace at the end of the patch command and all hunks will succeed.

January 23, 2018 at 9:34 PM • Reply »

OLEG SAYS: Thank you very much!
I already tried the new version of patch. worked well!
You also have to do “sudo apt-get install kernel-package” before compiling kernel.
Otherwise .deb-file will not be created.

January 24, 2018 at 12:19 PM • Reply »

MAX G SAYS: Most recent way to success (for me):

RPi 3 and the real time kernel – autostatic.com
https://autostatic.com/2017/06/27/rpi-3-and-the-real-time-kernel/

Index of /pub/linux/kernel/projects/rt/4.9/
https://www.kernel.org/pub/linux/kernel/projects/rt/4.9/

https://raw.githubusercontent.com/fedberry/kernel/master/usb-dwc_otg-fix-system-lockup-when-interrupts-are-threaded.patch
https://raw.githubusercontent.com/fedberry/kernel/master/usb-dwc_otg-fix-system-lockup-when-interrupts-are-threaded.patch

use an older version of the repository, accodring to the newest kernel found in kernel.org link above..
https://github.com/raspberrypi/linux/commits/rpi-4.9.y?before=6071d6022278d8b150e6427ba7b16624e0512358+70

January 28, 2018 at 4:40 PM • Reply »

LEAVE A REPLY

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

Post Comment