Slackware Live DAW: Compensating Latency

Overview of Latency and its Sources

Audio data is moved in "chunks", called buffers, which contain a number of audio samples. The buffer takes time to fill up, due to the fact that a system runs at a fixed sample rate. The size of the buffer is determined by the "Frames/Period" and "Periods/Buffer" settings given to the audio server.

E.g. A Frames/Period setting of 512, and Periods/Buffer of 3:
512 x 3 = 1536 Frames/Buffer (i.e. Samples/Buffer)
If the sample rate is set at 48000Hz, then the buffer fills in:
1536/48000 = 0.032 seconds, or 32 milliseconds

Larger buffers end up being audible as delay when playing a monitored instrument. For example, playing a chord on a MIDI piano and hearing it after a delay in headphones. The buffer settings can be reduced to find an acceptable amount of delay for playing, at trade-off of risk of xruns. Alternatively the buffer can be increased to allow more latency which gives more stable audio performance.

xruns *are audible glitches or pops that occur when the computer can't keep up with the task of filling up the buffer with samples. They occur more easily at lower buffer sizes and/or higher sampling rates, since it is more work for the CPU to write these smaller buffers at a quicker rate.*

In any case, the audio server knows how it is configured and can compensate for the known delay of the buffer size and sample rate. This compensation is done by delaying when recorded tracks are written to file, keeping all the timing of a multi-track recordings in sync.

However, there is still additional delay from the audio interface hardware (ADC's and DAC's) and remaining audio chain. These delays can be measured and input to the audio server to completely compensate signal latency. Several ways to achieve this will be outlined below.

Using QJackCtl and jack_iodelay

Slackware Live DAW come with QJackCtl to manage the jackd server, and starts this automatically at boot/login. This can be configured else-wise, but to work with the stock setup you can use "jack_iodelay" and QJackCtl to compensate your hardware latency.

Once QJackCtl is running the JACK server, open up a terminal and run "jack_iodelay", which is a utility that comes with JACK to measure hardware latency. Once jack_iodelay is running it will start looping and reporting its information in the terminal (you can exit at anytime with Ctrl+c).
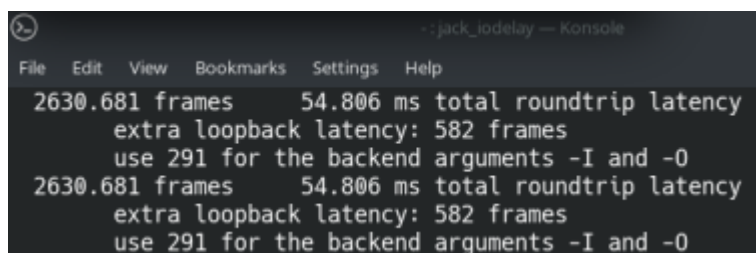
Open up the connection graph in QJackCtl and you will find a "jack_iodelay" program running. Connect its input to a physical input on your interface. Then connect the output of 'jack_iodelay' to a physical output.

The last step is to connect the physical output back to the physical input that jack_iodelay is using. Also don't use an output with speakers, the program uses a series of tones to measure latency and you probably don't want to hear them.

After all connections are made, jack_iodelay will report the full loop latency, in milliseconds, frames (samples), how much of the latency is due to hardware (it does this by subtracting the known latency from the total measured), and also recommends values to set for input and output ports.
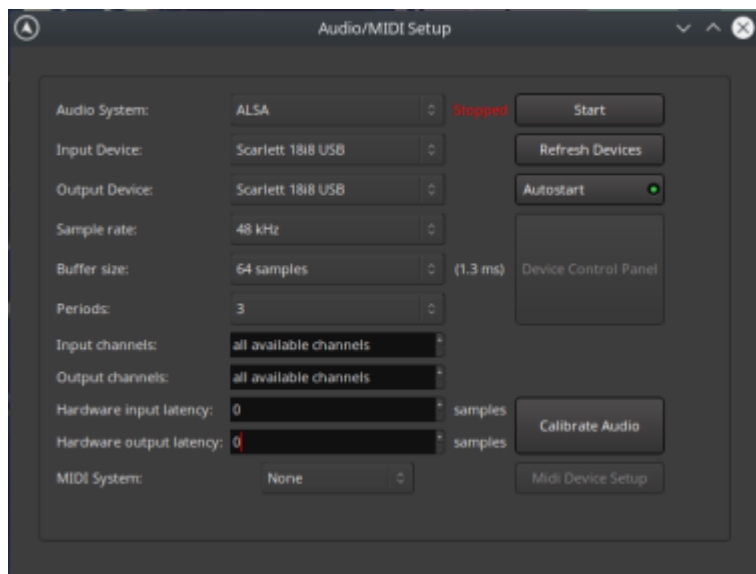

Quit jack_iodelay (Ctrl+c), stop the server in QJackCtl, and enter the "Setup menu". On the advanced tab there are entry spots for Input and Output latency. Enter the values from jack_iodelay here, then restart the server.
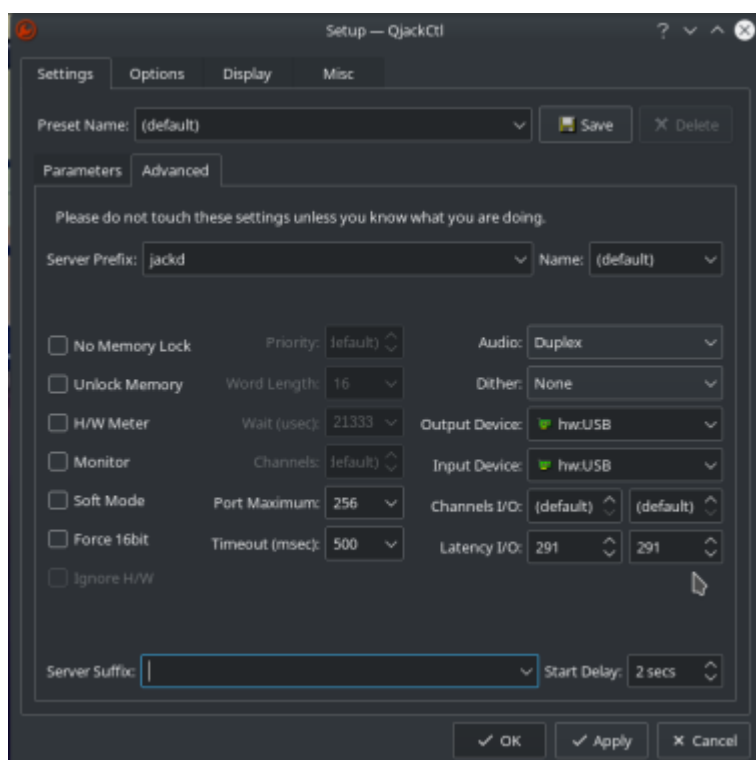

Now the jack server will be able to report the latency of its hardware ports as well, allowing programs to properly sync up overdubbed tracks when recording.

Using Ardour

Ardour comes with latency compensating tools built into its GUI (which are a little easier to use than first method). Note that Slackware Live DAW comes pre-configured to automatically start jackd/QJackCtl at startup, so running Ardour will just connect to this running instance. To use Ardour's built in tools you must start the server from Ardour. You can open QJackCtl from the tray in KDE Plasma and stop the server from there. Then launch Ardour and the program will prompt you to set up the audio server when you start a new session.

Last
update:
2020/08/05
21:04
(UTC)
howtos:multimedia:digital_audio_workstation:compensating_latency https://docs.slackware.com/howtos:multimedia:digital_audio_workstation:compensating_latency

Note that Ardour can use ALSA directly, or start its own JACK instance as well. In any case, select your audio interface for the Input and Output device and choose an acceptable sample rate, buffer size, and period size for your audio work. Then wire a loop back cable from an output to an input on your interface (the same as done in the first method). Once the hardware interface is looped, you can use the "Calibrate Audio" feature on Ardour's "Audio/MIDI Setup" page.
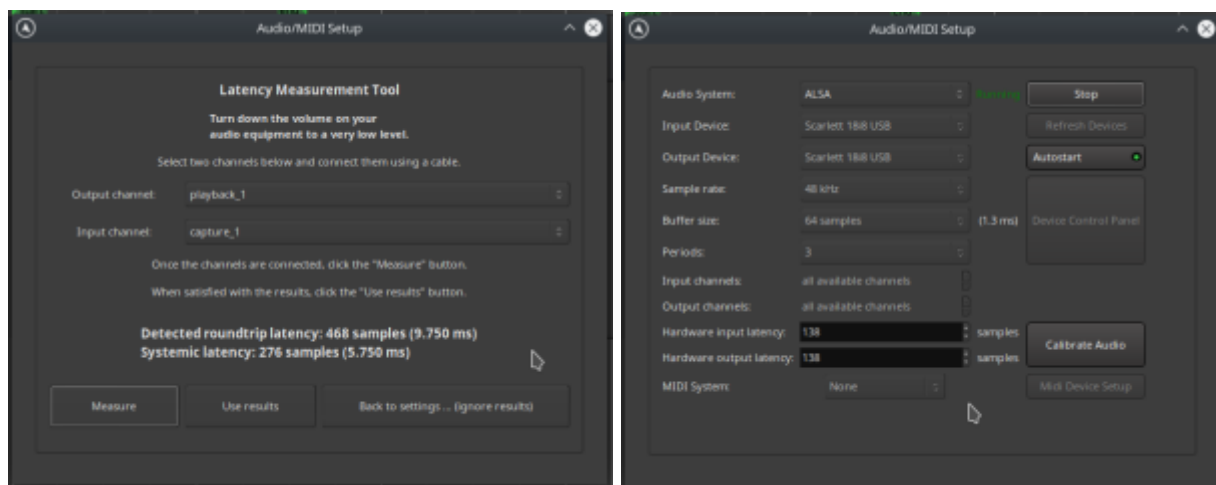
Select the appropriate ports that are looped, and click measure. The latency is measured, and the results are derived, the same as the first method (it is in fact running jack_delay in the back-end here).

You can set the latency compensation by clicking "Use Results", at which point the server will start.

An Alternative Test (and Visualizing Latency)

Both of the latency compensation techniques discussed in this article have used a loop back connection from the audio interface to send a signal through an output port and back to an input port. This return signal is delayed by the hardware latency, which can then be measured. You can also record this latency and see it for yourself by using a click track.

This example will use Ardour to show how to achieve this, since it its the primary DAW program that comes with Slackware Live DAW. Similar steps could be carried out with other recording programs.

Start a new recording session, without any latency compensation in place. Also loop an output back to an input on your audio interface, with the same technique as in the other latency compensating methods above.

Create a new audio track to record the click. Disconnect the input of this track. Then open up the "Audio Connections" Window. Route the "Click out" (Ardour's metronome) to the output. Then route the looped input to the track so you can record the click, through your hardware interface.
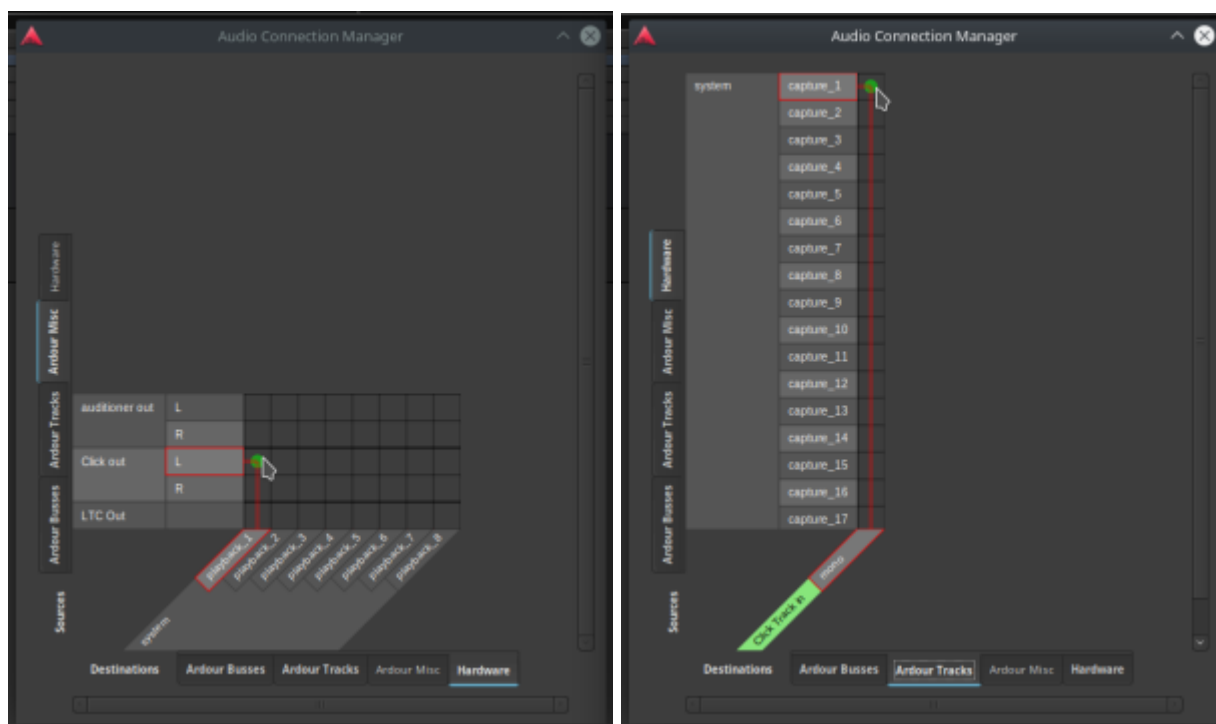
Then record a few seconds onto the track (with the metronome enabled).
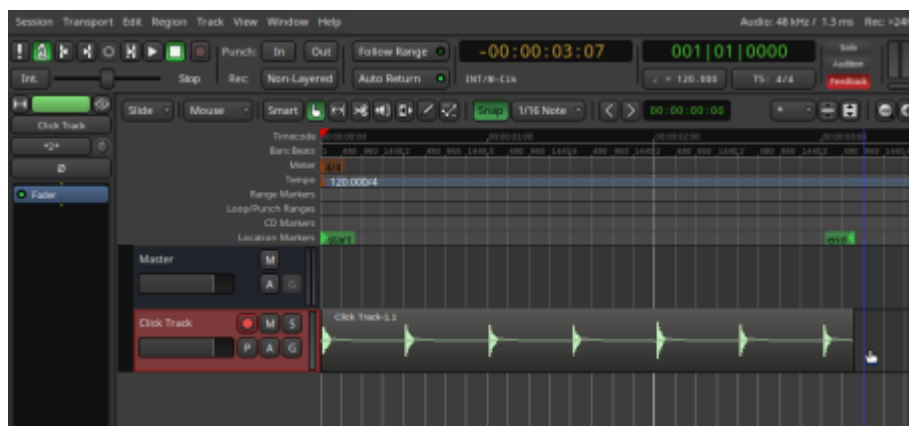

Zoom into a single impulse (Ctrl+Scroll-wheel). You will see that the impulse starts later than the grid-line, which is where the red play-head is in the image below. The metronome is sounded at each quarter note on the grid, but doesn't get recorded until later, due to hardware latency. For reference the image below was taken on a system running at a buffer size of 64, and 3 periods, and shows an offset of 511 samples.
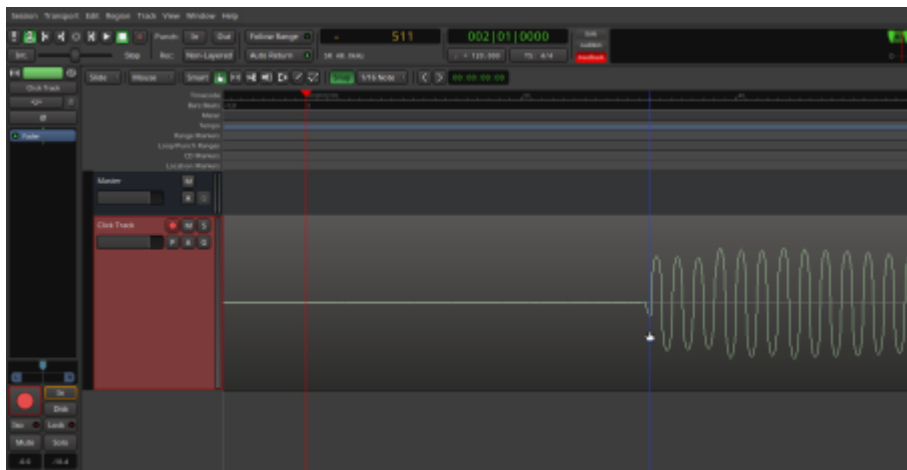
This works out to 511/48000 = 10.64 milliseconds of latency.

Stop the audio server and repeat the audio setup, but this time add the latency compensation. Once that is done, record another section of clicks, and zoom in again.
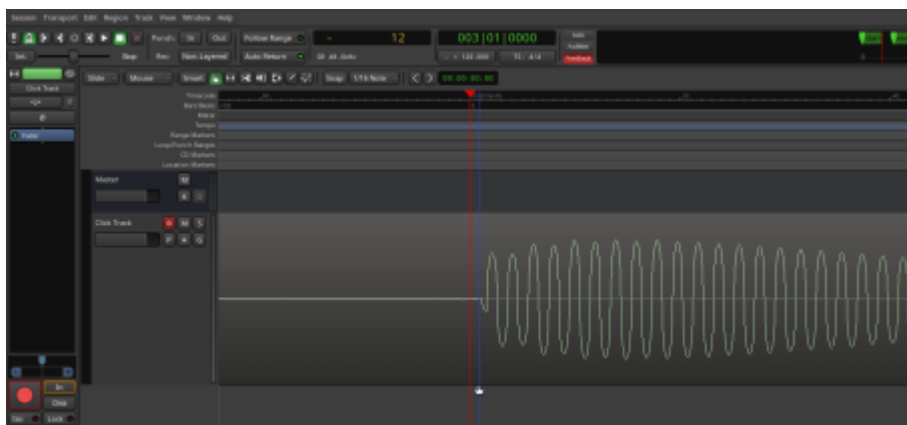
Now the impulse should line up on the grid line, or very close. This is because the audio server has compensated for the hardware delay, allowing the timing to be precise. In my case here my "observable latency" went down to 12 samples. This works out to a delay of 12/48000 = 0.25 milliseconds.

References

https://wiki.linuxaudio.org/wiki/jack_latency_tests
https://www.systutorials.com/docs/linux/man/1-jack_iodelay/
https://manual.ardour.org/synchronization/latency-and-latency-compensation/

Sources

- Originally written by Bob Funk
- Converted from latex sources using pandoc and sed by Eric Hameleers

howtos, multimedia, daw, audio, author 0xbf

From:
https://docs.slackware.com/ - **SlackDocs**

Permanent link:
https://docs.slackware.com/howtos:multimedia:digital_audio_workstation:compensating_latency

Last update: **2020/08/05 21:04 (UTC)**