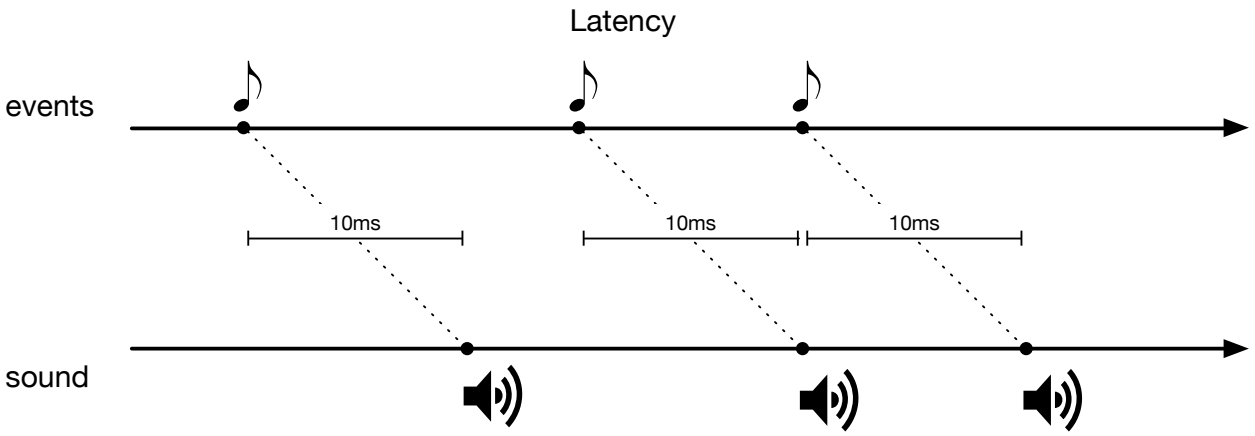
MIDI jitter might be ruining your live performance

A few months ago I set out to replace my MIDI guitar performance environment with a more flexible and intuitive solution. For various reasons I ended up using our own Eigenharp software, EigenD, even though I never intended to. One of the things that struck me over other solutions was that software synths felt much nicer to play. I always used to feel very disconnected from the musical performance and with my new setup this wasn't as much the case anymore. I never had time to analyze why exactly, until research I did last week revealed a critical lack of MIDI time-stamping support in most plugin hosts and DAWs.

Let's first get some basic terminology out of the way. Most timing aspects of electronic music can be described through latency and jitter.

What is latency?

Latency is a short constant delay that is introduced between the occurrence of an event and the actual perception of the audio.



Interestingly, musicians are extremely good at dealing with latency because it's predictable and already present all around us. The simple fact that audio transits through air introduces latency. Hence, anybody that plays in a band naturally compensates for latency, resulting from the different positions of other band members on stage or in the rehearsal room. Pipe organists are possibly the most extreme example of our capacity to deal with latency, due to delays of up to hundreds of milliseconds caused by the physics of the pipes and the spaces they're played in.

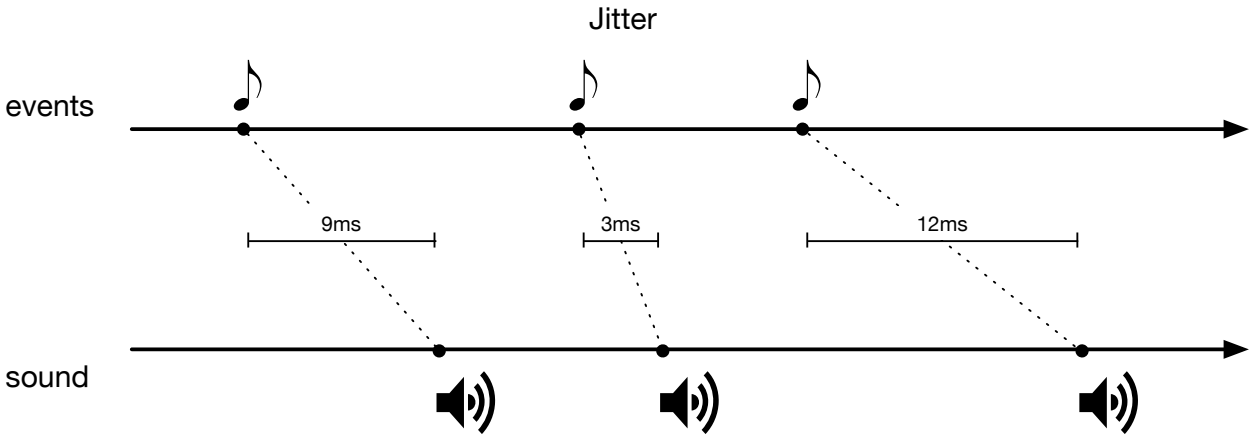
What is jitter?

Jitter on the other hand describes a deviation from a steady signal and makes evenly spaced events come out scattered.

Contrary to latency, jitter is very disconcerting for a musician as it's not constant and the timing of sound unpredictably changes from event to event: sometimes a note might be heard almost immediately but at other times it's heard with a varying delay that can't be

anticipated. At extreme levels, jitter makes it impossible to perform at all, but at lower variations of a few milliseconds, everything just feels off and you never connect with the music you're playing, without really being able to tell why. Academic studies have indicated that jitter becomes noticeable between 1 and 5ms, with a weighting around 2-3ms (Van Noorden 1975, Michon 1964, and Lunney 1974). Note that these tests only measured the perception of jitter while listening, not while performing. Intuitively I expect performing musicians to be even less tolerant to it.

Just as with latency, jitter can be introduced at many stages of your gear, it can be the clocking of an audio interface, the performance of a computer, the reliability of cables and much more. It turns out though that even if you're using good quality hardware that minimizes these timing problems with rock steady clocks, many software instrument hosts don't use MIDI event timestamps and cause the generated audio to be quantized to sample buffer boundaries, introducing jitter at the final step.



Let's wrap up the terminology with a quick recap of sample rate and buffer size.

Sample rate and buffer size

You probably know that digital audio is created by generating samples at a certain frequency, usually at 44.1kHz. This means that every second, 44100 sample values are calculated. These are then converted to the analogue domain by a D/A converter and result in sound coming from headphones or from loudspeakers.

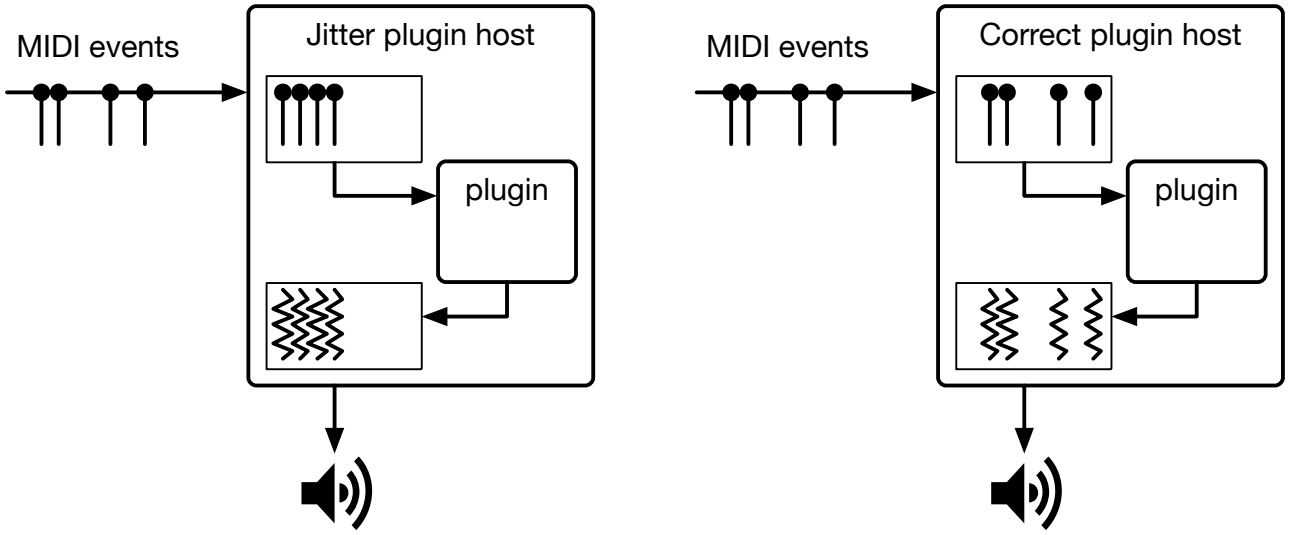
Since computers need time to calculate this data, software instrument hosts use one or several internal buffers of typically 256 or 512 samples. These buffers, combined with the latency of the hardware and drivers, is what people are talking about when they refer to the latency of software instruments. At a sample rate of 44.1kHz, a buffer of 512 samples provides the computer with 11.6ms (1000ms * 512 / 44100) to perform the calculations.

Essentially, the software is always running a bit behind to give it time to come up with the next batch of data, which the audio driver will periodically ask for. If this data isn't ready in time, you'll start hearing crackles and either need to increase the sample buffer size, reduce the sample rate or give the software instruments less work to do by, for example, reducing polyphony.

MIDI buffer timing

All this is pretty straightforward and intuitive, it gets a bit more complicated though when MIDI comes into play.

The timing of the MIDI data is completely unrelated to the audio stream, it's up to the host to correlate them. MIDI events merely arrive as reliably as possible at the host who subsequently has to pass them on to the audio plugin. Again, the host uses an internal buffer to store the MIDI data that arrived while the plugin was calculating a batch of samples. When it's time to calculate the next batch, the host simply passes the buffer of MIDI data to the software plugin so that it can calculate the corresponding audio.



It's here that additional MIDI jitter can set in. If the host doesn't remember when the MIDI events arrived while storing them in a buffer, the software plugin can only assume that they started at the beginning of the sample buffer it's generating and that they followed each-other, one by one, without any delays in between. Since everything is snapped to the beginning of a sample buffer, this effectively applies a quantization effect to the MIDI events with the duration of the sample buffer size. For a buffer of 512 samples at 44.1kHz, the audio is then quantized at intervals of 11.6ms.

Obviously, this means that the timing of the musical performance is lost and that the generated audio will unpredictably be timed differently for each event, hence introducing jitter.

Only some plugin hosts and DAWs behave correctly in this regard.

Real world implications

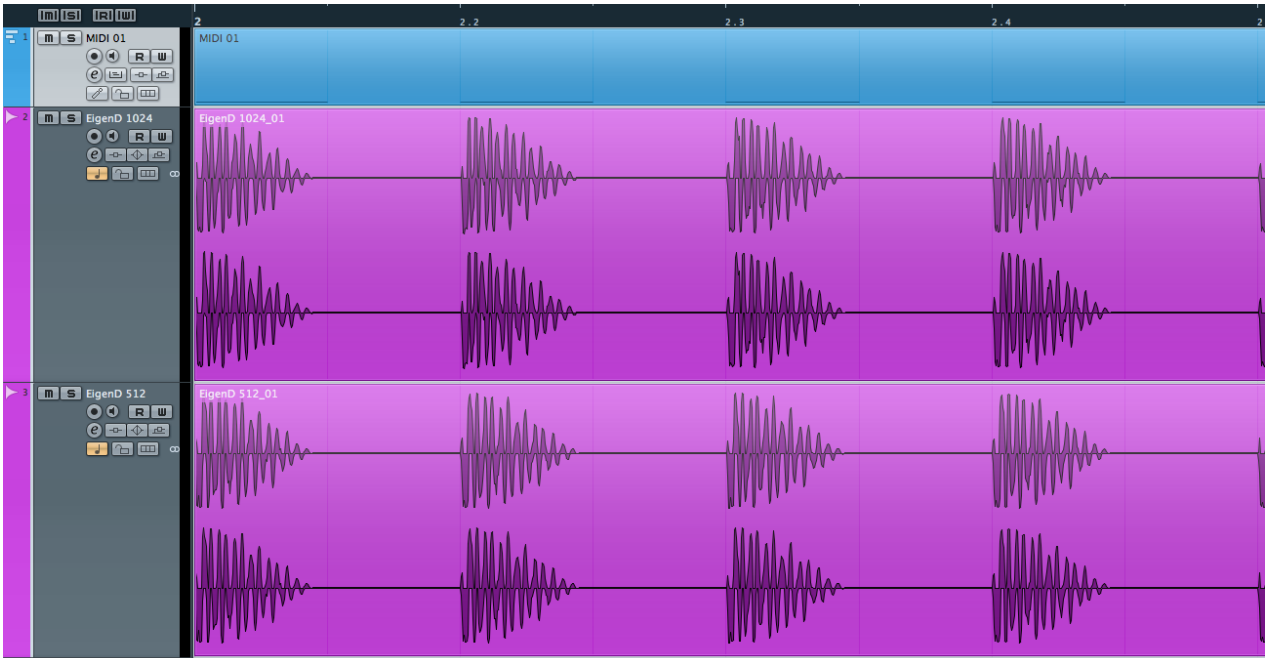
While it's interesting to theorize about the implications of MIDI buffer jitter, it's much more useful to look at a concrete example.

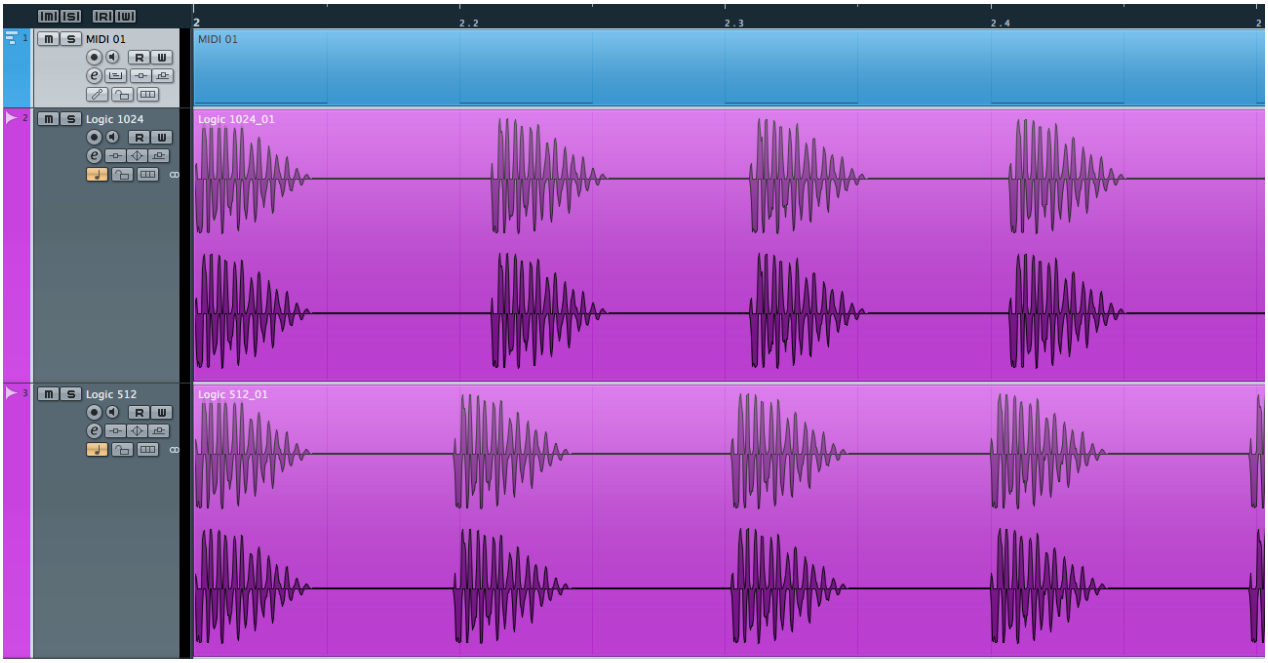
I set up a test environment in Cubase 6 on MacOSX, using the sequencer to play a constant stream of eight notes at 180BPM. The MIDI messages were sent over the local

IAC MIDI bus towards various plugin hosts. In each host I set up Native Instruments Absynth with a percussive snare sound. The audio output of the host was digitally sent through my Mac's built-in audio output towards my Metric Halo ULN-2 firewire interface over an optical cable, and finally recorded back into Cubase as an audio track. I tested four different sample buffer sizes for each host: 1024, 512, 256 and 128, all at 44.1kHz. When all the tracks were recorded, I aligned the start of the first recorded waveform with a bar line in Cubase so that all the other waveforms could be checked for timing by looking at the vertical lines of the beats.

Surprisingly I found that only about a third of the hosts behaved correctly and that many popular choices for live performance introduced MIDI buffer jitter. Note that this doesn't say anything about how these hosts behave when playing back MIDI from their internal sequencer through software instruments, this is handled differently and lies not within the scope of this analysis.

Below are two screenshots of the results in Cubase, the first of Eigenlabs EigenD's correct timing and the second of Apple Logic Pro with jitter.





You can see that the timing of the audio is all over the place in Logic, the beginning of each wave form is supposed to line up with the vertical beat lines. The screenshot shows that it is clearly deviating from these and based on the previous explanations, the maximum jitter with a buffer of 512 samples is 11.6ms. As sample buffer sizes are reduced, the jitter reduces also and a buffer of 128 samples would only have maximum 2.9ms of MIDI buffer jitter. The latter might be acceptable to some, but it's not considered beneath the perceptible limits.

The following table lists the hosts I tested and the categories they fall into. It might be possible that there are preferences that can be tweaked for each host. I approached them with default settings, doing the minimal amount of work to get the test environment to work. The actual project files I created can be downloaded as a zip file. If you are experienced with this behavior in one of the problematic hosts and know how to fix it, please let me know and I'll make the required changes here and add an addendum with host-specific instructions.

MIDI buffer jitter	Correct MIDI timing
	AudioMulch 2.2.2
AULab 2.2.1 (Apple)	
	Bidule 0.9727 (Plogue) **(*1)**
	Cubase 6.5.4 (Steinberg)
Digital Performer 8.01 (MOTU)	
DSP-Quattro 4.2.2	
	EigenD 2.0.72 (Eigenlabs)
Jambalaya 1.3.4 (Freak Show Enterprises)	
	Live 8.3.4 (Ableton)
Logic Pro 9.1.8 (Apple)	
Mainstage 2.1.3 (Apple)	
	Maschine 1.8.1 (Native Instruments)
Max/MSP 6.0.8 (Cycling 74)	
Rax 3.0.0 (Audiofile Engineering)	
	Reaper 4.3.1 (Cockos)
StudioOne 2.0.7 (Presonus)	
	Vienna Ensemble Pro 5.1 (VSL)
VSTLord 0.3	

Host specific instructions

1. **Bidule**: the MIDI section in the preferences dialog contains a ‘Reduce MIDI Jitter’ option that only kicks in for new MIDI device nodes that are either created manually or by reloading the project. Applying the preferences and keeping the host running without intervening on the graph doesn’t change this behaviour.

Final words

There are many technical challenges when playing live with software instruments. A low-latency, jitter-free, steadily clocked hardware setup with great audio converters is indispensable. However, on the software side it's also important to pick a solution that preserves the reliability of your hardware from end to end. Many of today's top choices for live plugin hosting introduce a non-neglectable amount of MIDI jitter that is bound to degrade your musical performance. It's important to be wary of this and select your software solution with the same care as your hardware.

Acknowledgements

This article was written and researched by Geert Bevin in December 2012. Thanks to Mike Milton, Roger Linn, John Lambert, Jim Chapman, Duncan Foster and Robin Fairey for proof reading and suggestions.

References

Lunney, H. M. W. 1974. “Time as heard in speech and music.” Nature 249, p. 592.

Michon, J. A. 1964. “Studies on subjective duration 1. Differential sensitivity on the perception of repeated temporal intervals.” Acta Psychologica 22, pp. 441–450.

Van Noorden, L. P. A. S. 1975. Temporal coherence in the perception of tone sequences. Unpublished doctoral thesis. Technische Hogeschool, Eindehoven, Holland.