WAM-studio, a Digital Audio Workstation (DAW) for the Web

Michel Buffa, Antoine Vidal-Mazuy

WAM-studio, a Digital Audio Workstation (DAW) for the Web

Michel Buffa
buffa@univ-cotedazur.fr
University Côte d'Azur, CNRS, INRIA
France

Antoine Vidal-Mazuy
antoine.vidal-mazuy@etu.univ-cotedazur.fr
University Côte d'Azur, CNRS, INRIA
France

ABSTRACT

This paper presents WAM Studio, an open source, online Digital Audio Workstation (DAW) that takes advantages of several W3C Web APIs, such as Web Audio, Web Assembly, Web Components, Web Midi, Media Devices etc. It also uses the Web Audio Modules proposal that has been designed to facilitate the development of inter-operable audio plugins (effects, virtual instruments, virtual piano keyboards as controllers etc.) and host applications. DAWs are feature-rich software and therefore particularly complex to develop in terms of design, implementation, performances and ergonomics. Very few commercial online DAWs exist today and the only open-source examples lack features (no support for inter-operable plugins, for example) and do not take advantage of the recent possibilities offered by modern W3C APIs (e.g. AudioWorklets/Web Assembly). WAM Studio was developed as an open-source technology demonstrator with the aim of showcasing the potential of the web platform, made possible by these APIs. The paper highlights some of the difficulties we encountered (i.e limitations due to the sandboxed and constrained environments that are Web browsers, latency compensation etc.). An online demo, as well as a GitHub repository for the source code are available.

CCS CONCEPTS

• **Software and its engineering → Abstraction, modeling and modularity**.

KEYWORDS

Web Audio, DAWs, plugin architecture, Web standards

ACM Reference Format:
Michel Buffa and Antoine Vidal-Mazuy. 2023. WAM-studio, a Digital Audio Workstation (DAW) for the Web. In *Companion Proceedings of the ACM Web Conference 2023 (WWW '23 Companion), April 30-May 4, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3543873.3587987

1 INTRODUCTION

Computer Assisted Music (CAM) is an ever-evolving field that uses computers to record, edit, and produce music. Digital Audio Workstations (DAWs) are specially designed software for CAM, allowing users to create and manipulate digital audio content,as well as MIDI content. Audio plugins are software modules that add additional features to DAWs, giving users greater flexibility and control over their music production. The CAM market became popular with the Atari ST (1985) and the Cubase DAW proposed by Steinberg (1989)[1]. Soon, the VST standard for audio plugins has been proposed (1997) and since then thousands of plugins have been developed, that can be used in the main native DAWs available on the market. On the Web platform, the first online DAWs appeared in 2008 and used the Flash technology. The first DAWs that used HTML5 and the Web Audio API for audio processing did not emerge until the period between 2015 and 2016. A DAW is a feature-rich software and therefore particularly complex to develop in terms of design, implementation, and ergonomics. It allows the creation of multi-track pieces by directly using audio samples (e.g. by incorporating into a track an audio file or by recording from a microphone or sound card input), mixing them, applying sound effects to each track (e.g. reverb, frequency equalization, or autotune on vocals), but also using tracks with virtual instruments (software recreation of a piano, violin, synthesizer, drum kit, etc.). The track is then recorded in MIDI format : as events corresponding to notes and additional parameters (such as the strength with which one pressed the keys of a piano keyboard, for example). These events allow the track to be played by requesting a virtual instruments to synthesize the signal. The DAW therefore allows for the playing, recording and mixing of both audio and MIDI tracks, edition of these tracks (copy/cut paste within a track or between tracks), the management of real-time audio effects and virtual instruments, mixing, and exporting the final project in a simple format (e.g. .wav or .mp3 file). In the native world, these effects and instruments are the "audio plugins" that extend the capabilities of standard DAWs. Since 1997, an important market for third-party plugin developers has developed. Four DAWs dominate the market (Logic Audio, Ableton, Pro Tools, Cubase), and the existence of several proprietary plugin formats complicates the task of developers. On the Web, the market is emerging, the technologies more recent, and few DAWs exist, mostly commercial. Nevertheless, an inter-operable plugin format called Web Audio Modules (WAM), currently undergoing active development exists and is backed by at least one commercially available online DAW.

2 THE WEB AUDIO API

Since 2018, the W3C Web Audio API has been a "candidate recommendation" (a frozen standard). It offers a set of Audio Node generators that process or produce sound. These nodes can be connected to form an "audio graph". The sound travels through this graph at a sample rate of 44100 times per second (default value, this can be changed) and undergoes transformations [3]. Some nodes are wave generators or sound sources corresponding to a microphone input or a sound file loaded into memory, others transform

WWW'23Companion,April30-May4,2023,Austin,TX,USA
https://doi.org/10.1145/3543873.3587987

[1] https://www.musicradar.com/news/tech/a-brief-history-of-computer-music-177299

the sound. Connecting these nodes in the browser is done through JavaScript and enables a wide range of different applications involving real-time audio processing [6]. Music applications are not the only ones requiring complex audio back-ends, and the API is also designed to meet the needs of games and other use cases, not only for computer music. The API comes with a limited set of standard nodes for common operations such as volume control, audio filtering, delay/echo, reverb, dynamic processing, spatialization, etc.

Usually, in multi-core processor systems, audio libraries/APIs split the work between a control thread and a render thread [7, 11, 12]. The Web Audio API is no exception and also uses this pattern: a render thread called the "audio thread" is solely responsible for rendering the audio graph and delivering the samples to the operating system so they can be played by the hardware. This thread has hard real-time constraints and has a high priority - if it fails to provide the next block of audio samples on time (128 samples by default in the case of the Web Audio API), there will be audible glitches. On the other hand, the control thread (generally running JavaScript code for the GUI, and making Web Audio API calls) manages all changes to the audio graph. It allows the user to connect/disconnect nodes and to adjust their parameters. Audio Nodes process the sound in the audio thread, and the algorithms used cannot be changed, only parameter changes are allowed. The recent addition (2018) of the AudioWorklet node provided a solution for implementing custom low-level audio processing running in the audio thread [6].

The nodes in the Web Audio API can be assembled into an "audio graph," which developers can use to create more complex audio effects or instruments. Here are some examples of audio effects that could be built this way: echo (DelayNode, BiquadFilterNode, GainNode), auto-wah (BiquadFilterNode, OscillatorNode), chorus (multiple DelayNodes and OscillatorNodes for modulation), distortion (GainNode, WaveShaperNode), synthesizers, samplers etc. Existing DSP code in other languages such as C/C++ or coded using Domain Specific Languages such as FAUST, can be cross-compiled to Web Assembly and run in a single AudioWorklet node. Over the years, lots of high-level audio effects and instruments have been developed[4]. However, it is often necessary to chain such audio effects and instruments together (for example, in a guitarist's pedalboard) and during music composition/production in DAWs, multiple effects and instruments are used. These are cases where the Web Audio API nodes are too low-level, hence the need for a higher-level unit to represent the equivalent of a native audio plugin [1]. For the Web Platform, such a high-level standard for "audio plugins" and "host" applications did not exist before 2015 [8]. Several initiatives have aroused and one emerged as a "community standard", which we detail in the following section.

3 WEB AUDIO MODULES

In 2015, Jari Kleimola and Olivier Larkin proposed the Web Audio Modules (WAM)[8], a standard for Web Audio plugins and DAWs. Soon later, Jari Kleimola was involved in the creation of ampedstudio.com, one of the first online DAWs using the WebAudio API. In 2018, the WAM initial project was further developed by researchers with the help of developers from the digital audio workstation industry, resulting in a more versatile standard [3]. Finally, the 2.0 version of Web Audio Modules has been released in 2021. This version aimed, in addition to setting a community standard (API, SDK), to bring more versatility to developers, more performance, to simplify access to plugin parameters, and facilitate integration in DAWs [5]. We will come back to this feature later, but WAM 2.0 uses an original design for handling the communication between plugins and host applications that do not rely on the low-level parameter management provided by the Web Audio API. The main reason for that is to allow high performances in the case where both a DAW and plugins are implemented as AudioWorklets. At the time the WebAudio API has been designed, AudioWorklets did not exist and some use cases could not be taken into account. Indeed, if the DAW is built using AudioWorklet nodes for processing audio, then some parts of the code run in the high priority / audio thread. Then, if a WAM plugin is associated with a given track in a DAW project, and if the plugin is itself built using an AudioWorklet node, it also has custom code running in the audio thread. The WAM framework has been designed to handle this particular case and will enable DAW/plugins communication without crossing the audio thread barrier. Let us take one example: while playing, a MIDI track sends notes to a virtual instrument plugin, and changes some of the parameters of this plugin at the frequency rate. Remember that a DAW can have multiple tracks associated to dozen of plugins, and each plugin can have dozens of parameters. The WAM framework detects this case, and will seamlessly use Shared Array Buffers and a ring buffer, without crossing the audio thread barrier. No need to send events from the control/GUI thread, that would have been mandatory if the Web Audio API audio node parameter management was used. To sum up, the WAM framework streamlines the creation of plugins and host applications and enables highly efficient communication between hosts and plugins.

4 RELATED WORKS: COMMERCIAL AND OPEN SOURCE DAWS

The first online DAWs based on the WebAudio API appeared in 2015-2018 and are still available today: Audiotools[9], Bandlab, AmpedStudio, Soundtrap [10], Soundation. These DAWs have in common the fact that they are commercial, closed source and received very little academic publication. They facilitate remote collaboration on musical projects, and have gained increased attention during the COVID-19 pandemic (2021-2022). The mode of collaboration varies (synchronous like Google Docs or asynchronous through sharing links), as well as the communication tools provided (chat, video conferencing), but all these DAWs have the classic features: audio and MIDI recording, track editing, mixing, support for effects and virtual instruments, etc. Some have been designed mainly for desktop computers while others are particularly adapted to mobile devices.

They mainly differ in terms of ergonomics : some of these applications are aimed at a very wide audience and have focused on simplicity (BandLab, SoundTrap) while others have chosen a more "professional" and sober look and feel, like AmpedStudio. Audiotools is more a "virtual studio" than a pure DAW and finds its market mainly in the education domain. Implementation details of these applications are unknown as the source code is not publicly accessible. However, through limited academic publications and presentations/interviews, it is known that AmpedStudio uses a

		bandlab	ampedstudio	soundtrap	soundation	arpeggi	audiotool
Target	Desktop	X	X	X	X	X	X
	Mobile (native or responsive webapp)	X		X		X	
	Oriented to users familiar with Computer Music		X	X			
	Premium (free, with premium functionalities)	X	X	X	X		X
functionalities	Basic DAW features (recording, editing, mixing etc.)	X	X	X	X	X	X
	Audio effects and virtual instruments	X	X	X	X	X	X
	Commercials plugins support (not documented)	X	X	X	X		
	Open plugins support (WAM)		X				
	Cloud (saving project , loading, sharing, etc.)	X	X	X	X	X	X
	Collaborative (asynchronous)	X	X	X	X	X	X
	Collaborative (synchronous)			X			X
	Call in app		X	X			X
Technologies	Using AudioWorklet	X	X	?			X
	C++/wasm audio engine	X	X				
	Native mobile application	X					
	Blockchain					X	
Open Source							

Figure 1: Comparison table of the main online DAWs on the market

cross-compiled C++ engine and AudioWorklets, while Soundtrap has primarily utilized the standard Web Audio API nodes. BandLab is supposed to have a core written in C++, and relies also on AudioWorklet, with possibly a specific mobile version. Audiotools runs also in AudioWorklet node(s) with a flash port of the audio engine, effects, instruments... Recently, a new DAW named Arpeggi.io has emerged, which highlights its use of the blockchain technology (for tracking who, when, and how sounds and music are used) and NFTs for monetizing creations. These DAWs differ also in the way they manage audio effects and instruments. It is obvious that some support external plugins: AmpedStudio supports the WAM standard -developers from the company contributed to the proposal- and the web site has a shop for "enabling" premium plugins. The other commercial online DAWs seems to use a proprietary format, while integrating some third party ports of native plugins (Soundtrap proposes the famous "Autotune" plugin made by Antares). Propellerheads, a company well known for its native DAW Reason, also published web versions of its Europa synthesizer plugin (available now in AmpedStudio and in Soundation). So far, AmpedStudio is the only DAW that supports an open plugin standard: Web Audio Modules. Figure 1 illustrates the similarities and differences between these commercial online DAWs.

Gridsound is an open source DAW, that has been developed since 2015[2]. It supports audio and MIDI and is a fully functional DAW. It does not support plugins but comes with a set of audio effects and virtual instruments whose format is specific. There are also popular JavaScript libraries for developing a multi-track player/recorder, such as wavesurfer.js or waveform-playlist[3], and some audacity like audio buffer editors such as AudioMass[4]. None of these open source initiatives use AudioWorklets and low-level processing, nor take care about optimizing DAW/plugin communication, preserving audio thread isolation, nor use external plugins. This is the main reason why we developed Wam-Studio: as a demonstrator of these techniques.

5 WAM-STUDIO DESIGN AND IMPLEMENTATION

Wam-Studio is an online tool for creating audio projects that you can imagine as multi-track music. Each track corresponds to a different "layer" of audio content that can be recorded, edited, or just integrated (using audio files for example). Some track can be used to control virtual instruments: in that case we record the sound that is generated internally by these virtual instruments (and played using a MIDI piano keyboard, for example). Tracks can be added or removed, played isolated or with other tracks. They can also be "armed" for recording, and when the recording starts, all other tracks will play along, while the armed track will record new content.

5.1 The tracks

A DAW track is a container for audio-related data that comes with an interactive display of these data, editing and processing facilities and few default parameters such as volume and left/right panning. Figure 2 shows an isolated audio track with the wave-

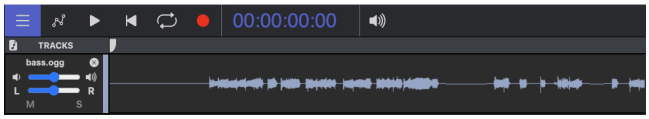

Figure 2: A track in the DAW GUI

form display of the associated audio buffer, and the default track controls/parameters on the left side (mute/solo, volume, stereo panning). As many tracks can be rendered, scrolled during playback, zoomed in/out and edited, we used the pixi.js library to handle drawing and interactions. This library uses gpu accelerated webgl rendering, and provides many features for handling multiple layers over a single HTML5 canvas. As shown in Figure 4, each track can also be associated with a set of plugins for adding audio effects or for generating music (in the case of instrument plugins). Figure 3 shows the audio graph that corresponds to the audio processing chain of an audio track. The sounds goes from left to right: first the "track player/recorder/editor" is implemented as an AudioWorklet node, using custom code for rendering an audio buffer, then the

[2]https://gridsound.com/. See also the GridSound video presentation at ADC 2021 conference: https://www.youtube.com/watch?v=ejTtENwRxnA
[3]https://github.com/naomiaro/waveform-playlist
[4]https://audiomass.co/

output signal has its gain and stereo panning adjusted, then we have another AudioWorket node for rendering the volume in a canvas (vu-meter), and at the end we find a chain of WAM plugins for adding audio effects.

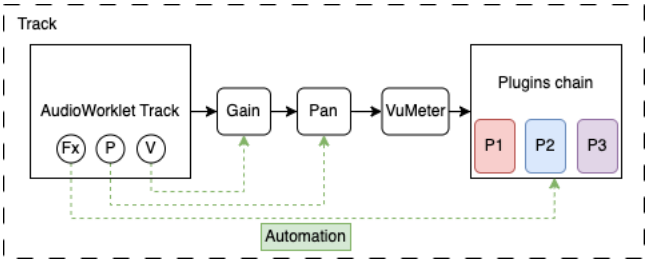

Figure 3: Audio graph of an audio track in Wam-Studio

There are two types of tracks: **Audio tracks** : they contain recorded audio, such as a vocal take, a guitar recording, or any other type of audio signal, that are generally rendered graphically as waveforms that represents the actual signal stored (as a set of sound samples). These tracks can be edited, processed and mixed by copying/cutting and pasting audio samples in the audio buffer associated to the track. Audio track's output can be further processed by a chain of audio effects. **MIDI tracks** : they contains MIDI data (the pitch -the note an instrument would play, velocity, and duration). MIDI data does not contain audio information but are rather used to control virtual instruments such as synthesizers, samplers, drum machines, etc. MIDI tracks can be edited, in that case, it is possible to change the MIDI events stored in the track in a matrix-like display. MIDI tracks are generally associated with one or more virtual instruments. The MIDI events can target all track instruments or some specifically. The tracks are usually organized vertically within the DAW's graphical user interface and can be separately managed in terms of volume, stereo positioning, effects processing, and automation of effects or instruments (as shown in Figure 4).

Figure 4: A plugin chain associated to the selected track

Each track output is connected to a "master track" where it is possible to adjust globally the volume and panning of the final mix. Like for any other tracks, it is also possible to associate a chain of audio effects to the Master track (adjusting final dynamics, frequencies). All audio effects and virtual instruments are WAM plugins in the case of our DAW. This design gives an extensive degree of control and adaptability and enables users to blend and manipulate the sound of each track with high precision and sophistication, thus making it easier to create intricate audio productions. A DAW track is also capable of "rendering" the track into an audible audio signal (when the track is being played). When one presses the play button of the DAW, all the tracks are rendered simultaneously, and the final output signal is what is called "the mix". DAWs also have an "offline rendering" option, for exporting the final mix as a file, on local hard disk or on the cloud (using the OfflineAudioContext[5] provided by the WebAudio API).

5.2 Track core implemented as an AudioWorklet processor

Let us focus for the moment on an Audio track. The Web Audio API provides the AudioBufferSourceNode that is a container for an audio buffer. This standard node is enough to play any audio buffer stored in memory. However, we will explain why this is not sufficient for a high performance DAW... Remember what can be done in DAWs while a multi-track piece is being played (or recorded): parameter automation! This means interpolating plugin and track parameters at the frequency rate, 44100 times per second. Moreover, these automated parameters can encompass any characteristic of any plugin associated with any track. It is a substantial amount of information that can be exchanged between the DAW and WAM plugins, making performance a crucial aspect that requires careful consideration. By designing our own low-level audio player (instead of using the standard AudioBufferSourceNode), we gain total control over the playback/recording behaviour of each track. We designed each audio track core as an AudioWorklet processor, more precisely, as an instance of a subclass of the WamProcessor class provided by the Web Audio Modules SDK, that inherits from the AudioWorkletProcessor class. According to the API, each processor implements a process method that will be called at the sampling rate. Inside this method, we output the sound samples and render the buffer, but we can also send data to WAM plugins. The WAMProcessor class we inherit from provides methods for handling DAW/plugins communication efficiently. With the WebAudio API, AudioWorklets are comprised of two components, reflecting the multi-threaded nature of the environment: the AudioWorkletNode and the AudioWorkletProcessor. The WAM framework extends them into the WamNode and the WamProcessor classes, which serve as the main thread and audio thread of WAM powered software, respectively. WAM-Studio is a WAM host application, and its tracks are implemented as sub-classes of WamProcessor. The WAM plugins associated to these tracks can also have their DSP core implemented as subclasses of WamProcessor, but this is not mandatory. Some WAM plugins implementations utilize a single node, while others utilize a subgraph consisting of a combination of built-in web audio nodes and custom audio nodes, as abstracted through the WamNode and WamProcessor interfaces. This abstraction enables seamless interoperability between a WAM hosts and plugins, regardless of the underlying implementation, and across all threads.

[5]In contrast with a standard AudioContext, an OfflineAudioContext doesn't render the audio to the device hardware; instead, it generates it, as fast as it can, and outputs the result to an AudioBuffer

In other words : if a track is a WamProcessor and if plugins are also WamProcessors, their DSP code runs in the audio thread and highly optimized communication can be achieved: Shared Array Buffers can be used for DAW/Plugin communication. If a plugin is not a WamProcessor, then the DAW code that "talks" with it remains unchanged and the underlying implementation will be suboptimal and will involve crossing the thread barrier. Handling multiple plugins in a chain: The WAM framework employs a singleton object, referred to as the WamEnv, which is attached to the global scope of the audio thread and serves as a mediator for interactions between hosts and processors/plugins. Processors/plugins are structured into WamGroups, managed by the WamEnv, where each group comprises plugins created by a specific host. Considerable effort was exerted to reduce the number of assumptions made by the API regarding host implementation, with the WamEnv and WamGroup being the only objects within the WAM system that are anticipated to be supplied by the host. WamEnv is allocated when the DAW is created, and WamGroup instances are associated with each track.

5.3 Managing plugin chains

Plugin chains are handled using a special WAM plugin that acts also as a "mini host". We called it the WAM pedalboard [1]. It connects to a plugin server that sends back the list of plugins available as a JSON array of URIs (a WAM plugin can be loaded simply using a dynamic import and its URI, see[5]). From this list of URIs, WAM plugin descriptors are retrieved, that contain metadata about the plugins: name, version, vendor, thumbnail image URI, etc. When the pedalboard plugin is being displayed in the DAW, it is empty, and plugins can be added to the processing chain, removed, re-ordered, and their parameters can be set. Any configuration can be saved as

Figure 5: the WAM Pedalboard that manages plugin chains associated to DAW tracks

a named preset (e.g. "guitar crunch sound 1"). Presets can be organized into banks ("rock", "funk", "blues"). The management of the organization and naming of banks and presets is the responsibility of the pedalboard plugin. The parameters exposed by this plugin correspond to the entire set of parameters of the active preset (that is, the sum of the parameters of the plugins associated in the preset), and can be automated by the DAW. The Web Audio Modules API allows for sending events to the DAW when changes occur in the plugin configuration (addition or removal) and the menu provided in the DAW for selecting automatable parameters is automatically updated. All WAM plugins implement getState/setState methods to serialize/deserialize their state. When a project is saved, the state of each track, audio buffers, etc. are saved, as well as the state of the plugin configuration.

5.4 Recording process, dealing with latency

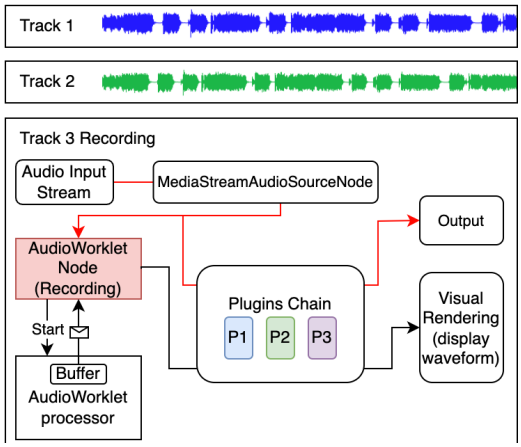

Figure 6: Track recording process in WAM-Studio

The recording process using a DAW is more constrained and structured compared to using a simple recording app like a memo recorder, as we must take into account different kind of latencies. The input latency is the time between when an audio signal is captured by the input device and when it is processed by the audio context. It depends on the OS, the configuration of the input device, the processing time of the audio system, and the buffer size used by the audio context. The output latency represents the time between when a sound is generated and when it is actually heard by the user. The sum of these two latency represents the round-trip latency and can be measured using an external recording device with two microphones: one at the source of the physical sound (for example, on the body of a guitar plugged in a sound card), and one in front of the speakers. Hit the body of the guitar, record the output and compare the input and the output signals. We did such measurements in the past[2] and showed that the round-trip latency using MacOS with common sound cards available on the market was around 19-23ms with an audio buffer of 128 samples (on Windows this was much bigger, 50-75ms). The WebAudio API exposes two properties of the AudioContext that could have been useful. The baseLatency property represents the number of seconds the AudioContext uses for its internal processing once the audio reaches the AudioDestinationNode (final output of the audio graph). On Chrome it represents the minimum guaranteed latency that an audio context will add to the input audio stream before it is played through the output, and can be calculated as the audio buffer size divided by the sample rate. It is a rough approximation of the real input latency, while on Firefox it is always zero (as Firefox processes the audio directly in the audio callback[6]). The

[6]See Paul Adenot blog post https://blog.paul.cx/post/audio-video-synchronization-with-the-web-audio-api/

outputLatency property represents the number of seconds between the audio reaching the AudioDestinationNode (conceptually the audio sink of the audio processing graph) and the audio output device. In February 2023 it is not implemented in all major browsers. This situation makes these properties nearly useless, as we will see. Figure 6 shows the audio graph involved during the track recording process: we need first to obtain a MediaStream from the user's microphone or from a sound card input using the MediaDevices API. Then, we can use a Web Audio API MediaStreamSourceNode in the audio graph, passing the MediaStream to its constructor. This is how we expose a live audio stream to the Web Audio API. In a second step, we need a way to record this stream in a buffer. This can be done using the W3C MediaRecorder API, or using low-level solutions using an AudioWorklet. The MediaRecorder solution has for advantage its simplicity but all tests we conducted showed that this is an unreliable solution as the time it takes for allocating audio buffers and starting recording is unpredictable, even when we conduct latency measurements/calibration, as explained later on. We ultimately decided on a low-level, precise solution that uses a WamProcessor to record the sound samples that are received. This solution was crucial, as it was necessary to have full control over the buffer in order to expand it during extended recording sessions. When recording a guitar track, for example, other tracks such as drum and bass tracks can be being played simultaneously, and the guitar sound is processed by a chain of plugins (i.e guitar amplifier simulation, EQ, delay,etc.). The processed recorded guitar sound must be heard real-time in sync with the other tracks and free from any noticeable latency while being played (and recorded). This is why the overall round-trip latency should be as low as possible. We represented it with the red path in Figure 6. Using a 128 sample buffer size, the round-trip latency of 23ms we measured on MacOS[2], even with pro guitar players, was comfortable and comparable with what we can achieve using native applications in similar use cases (guitar, Logic Audio, guitar amp sim plugin, audio buffer size 128). The recording process requires also proper allocation and management of the track audio buffer, as well as continuous storage of the sound samples coming from the input stream. Here, the input latency is important as the recording process may be delayed due to hardware and software configurations (audio driver, etc.), but also the buffer allocation should be done with care: pre-allocate it before even pressing the recording button, use a ring buffer and start recording as soon as the track is "armed" (DAWs generally provide an "arm" button to select tracks that will be recorded), then store a timestamp when the recording really starts. To ensure that the recorded guitar track plays back in sync with the other tracks, we do some "latency compensation" i.e shift back in time the buffer content, depending on the input latency. Do you remember that we do not know its accurate value? The problem is that it is rather complicated to find this number in a program running, for a variety of reasons (again, see Paul Adenot blog post on footnotes). The solution we use consists in calibrating the DAW along with the hardware used, prior to recording. This process involves the use of a microphone to record a pre-generated sound emitted by the DAW and determining the time elapsed between the emission of the signal and its recording. This value is then used for latency compensation. The AmpedStudio DAW proposes a similar approach for calibration in its setting menu. Soundtrap authors explained that they rather use lookup tables with entries for the most common OS/soundcard pairs, calibrated by hand, and hard coded in the code[10]. They then use some heuristics to guess the current configuration (checking the OS, guessing the sound card model looking at the number of inputs/outputs exposed using the MediaDevice API, etc.). In WAM-Studio, latency compensation is a work in progress: for the moment we do it manually by entering a value measured by an external utility software[7] we developed, that uses the calibration process we just described (emit a sound, record, compare input and output).

6 CONCLUSION

As of today, WAM-Studio is an open source example of a DAW using advanced features such as external plugin support, high performance DAW/plugins communications, audio track playing, recording and editing. As a demonstrator, it shows the capabilities of the Web Audio API, and how the Web Audio Modules framework can be used to ease the development of complex web based audio applications with support for external plugins. Midi tracks support is still a work in progress[8] and development is active. The source code can be found on a GitHub repository[9], and the application is available online[10].

REFERENCES

[1] Michel Buffa, Pierre Kouyoumdjian, Quentin Beauchet, Yann Forner, and Michael Marynowic. 2022. Making a guitar rack plugin -WebAudio Modules 2.0. In *Web Audio Conference 2022*. Cannes, France. https://hal.inria.fr/hal-03812948

[2] Michel Buffa and Jerome Lebrun. 2017. Real time tube guitar amplifier simulation using WebAudio. In *Proceedings of the Web Audio Conference (Queen Mary Research Online (QMRO) repository. http://qmro.qmul.ac.uk/xmlui/handle/123456789/26089)*. Queen Mary University of London, London, United Kingdom. https://hal.univ-cotedazur.fr/hal-01589229

[3] Michel Buffa, Jerome Lebrun, Jari Kleimola, Oliver Larkin, and Stephane Letz. 2018. Towards an open Web Audio plug-in standard. In *WWW2018 - TheWebConf 2018 : The Web Conference, 27th International World Wide Web Conference*. Lyon, France. https://doi.org/10.1145/3184558.3188737

[4] Michel Buffa, Jerome Lebrun, Shihong Ren, Stéphane Letz, Yann Orlarey, Romain Michon, and Dominique Fober. 2020. Emerging W3C APIs opened up commercial opportunities for computer music applications. In *The Web Conference 2020 - DevTrack*. Taipei, Taiwan. https://doi.org/10.13140/RG.2.2.16456.19202

[5] Michel Buffa, Shihong Ren, Owen Campbell, Jari Kleimola, Oliver Larkin, and Tom Burns. 2022. Web Audio Modules 2.0: An Open Web Audio Plugin Standard. In *WWW '22: The ACM Web Conference 2022*. ACM, Virtual Event, France, 364–369. https://doi.org/10.1145/3487553.3524225

[6] Hongchan Choi. 2018. Audioworklet: the Future of Web Audio.. In *Proceedings of the International Computer Music Conference*. Daegu, South Korea, 110–116.

[7] François Déchelle, Riccardo Borghesi, Maurizio De Cecco, Enzo Maggi, Butch Rovan, and Norbert Schnell. 1999. jMax: an environment for real-time musical applications. *Computer Music Journal* 23, 3, 50–58.

[8] Jari Kleimola and Oliver Larkin. 2015. Web Audio Modules. In *Proceedings of the Sound and Music Computing Conference*.

[9] Bojan Lazarevic and Danijela Scepanovic. 2010. Unrevealed Potential in Delivering Distance Courses: the Instructional Value of Audio. *eLearning & Software for Education* (2010).

[10] Fredrik Lind and Andrew MacPherson. 2017. Soundtrap: A collaborative music studio with Web Audio. In *Proceedings of the Web Audio Conference*. Queen Mary University of London, London, United Kingdom.

[11] James McCartney. 2002. Rethinking the computer music language: Super collider. *Computer Music Journal* 26, 4, 61–68.

[12] Miller Puckette. 1991. FTS: A real-time monitor for multiprocessor music synthesis. *Computer music journal* 15, 3, 58–67.

[7] https://github.com/micbuffa/WAlatencyCompensation

[8] We wrote some demonstration of MIDI track playback using WAMs on the WAM tutorial site: https://wam-examples.vidalmazuy.fr/

[9] https://github.com/Brotherta/wam-refont

[10] https://wam-openstudio.vidalmazuy.fr/