GPU audio

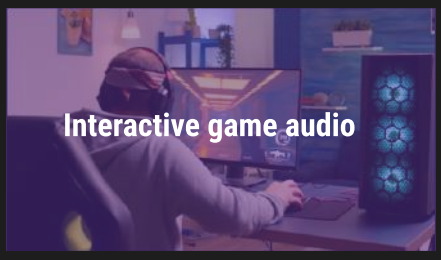
Unlocking the GPU for Real-Time Audio

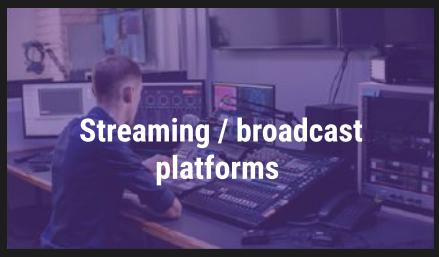
Contents

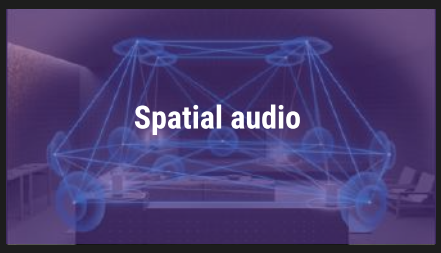
October 2022

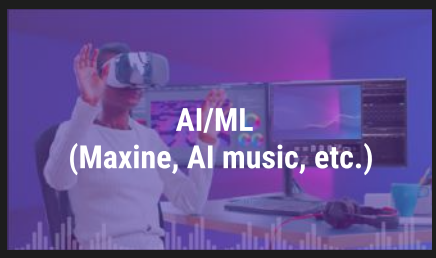
Market Needs & Challenges

Next Gen Audio needs a new standard









GPUs offer virtually unlimited power & scalability to digital audio compared to CPUs, but the problems are as inherent as the possibilities

There are three major challenges for GPU Audio processing

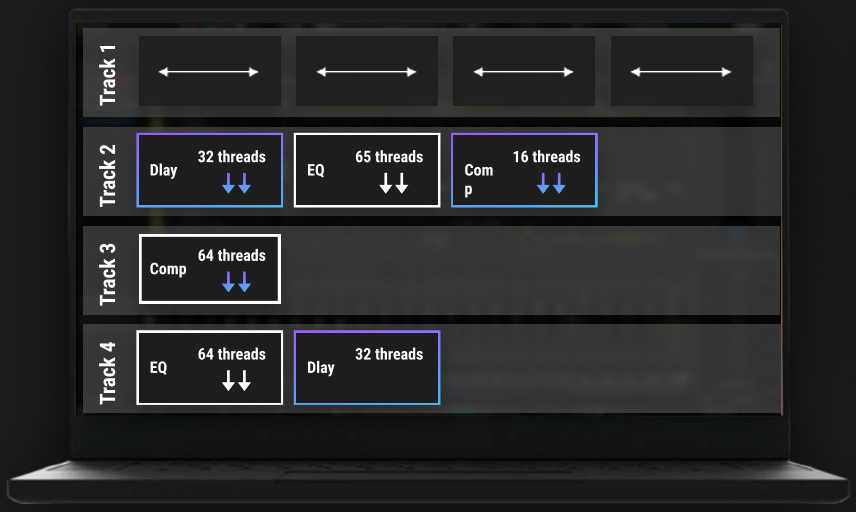
1 Parallelism and Heterogeneity

2 Multiple tracks and effects

3 Data transfer between CPU and GPU

Challenge 1: Parallelism and Heterogeneity

Parallelizing across tracks is not possible due to different thread counts and different effects



User's goals

Run loads of tracks with loads of effects

Get 1ms latency

Challenge 2: Large number of tracks and effects with different parameters

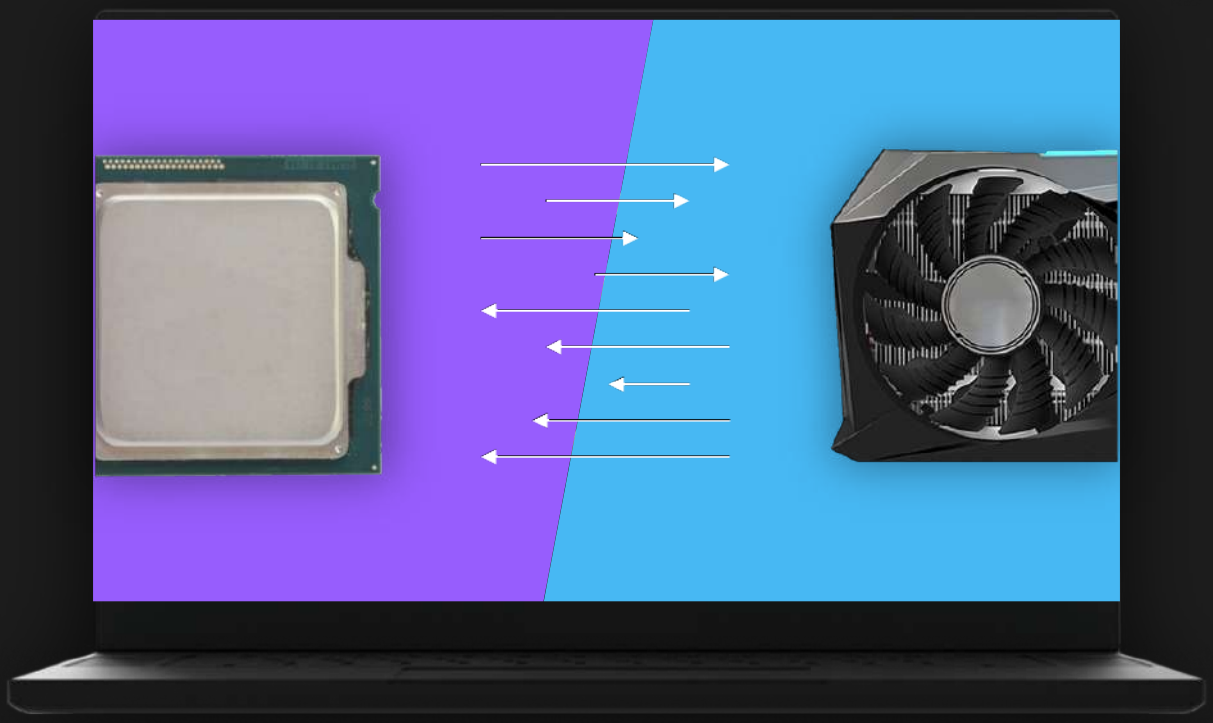
User's goals

Run loads of tracks with loads of effects

Get 1ms latency

Challenge 3: Data transfer between CPU and GPU

1000 asynchronous data transfers in each direction of 96 samples each just do not work



User's goals ✕

Run loads of tracks with loads of effects

Get 1ms latency

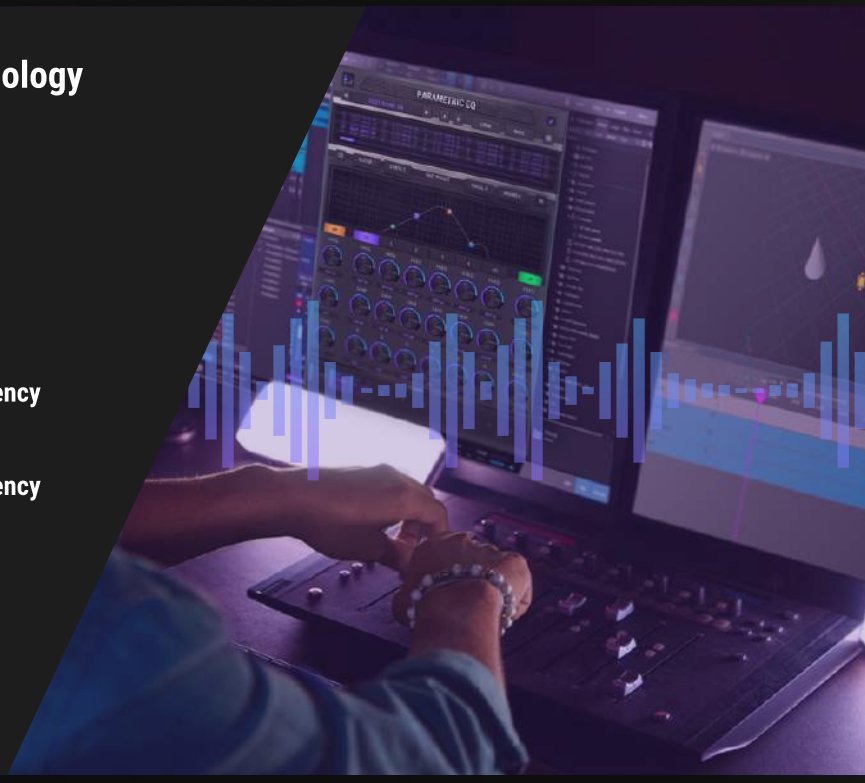
October 2022

Introducing GPU Audio Solution

GPU Audio: the world's first novel technology enabling real-time audio DSP on GPUs

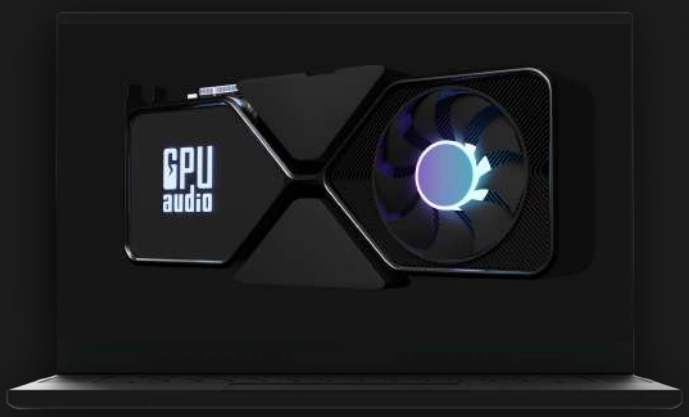
GPU Audio: the world's first novel technology enabling real-time audio DSP on GPUs

- Real-Time high performance DSP
- 1ms latency, regardless of channel count or effect instance count
- Powerful parallel processing, without added latency
- Powerful parallel processing, without added latency
- Network ability and scalability
- Audio processing over Ethernet, wifi6 or 5-6g connections



GPU Audio bridges the gap between Pro Audio and Next Gen Tech Standards

Features of our solution open up a world of high-performance audio and production to emerging technologies

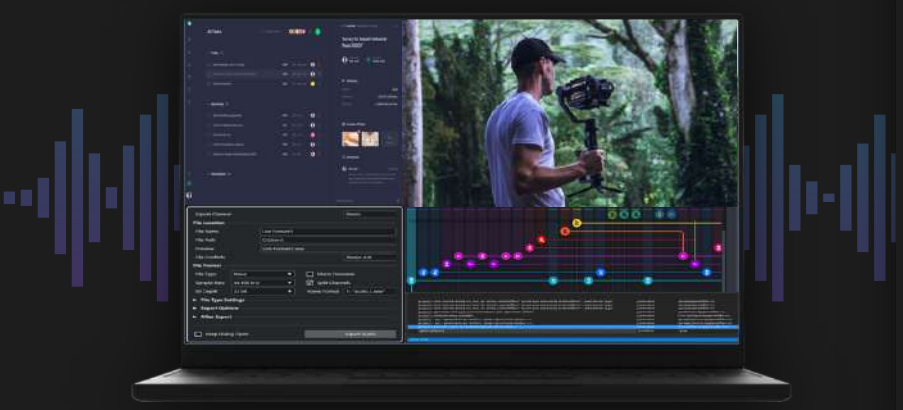


- AI/ML
- Web3 experiences
- Interoperable collaboration platforms like NVIDIA's Omniverse
- Spatial audio computing and simulation
- Next Gen DAW and music production software

Bounce mode & rendering accelerated by GPU

GPU audio

GPU Audio: the world's first novel technology enabling real-time audio DSP on GPUs

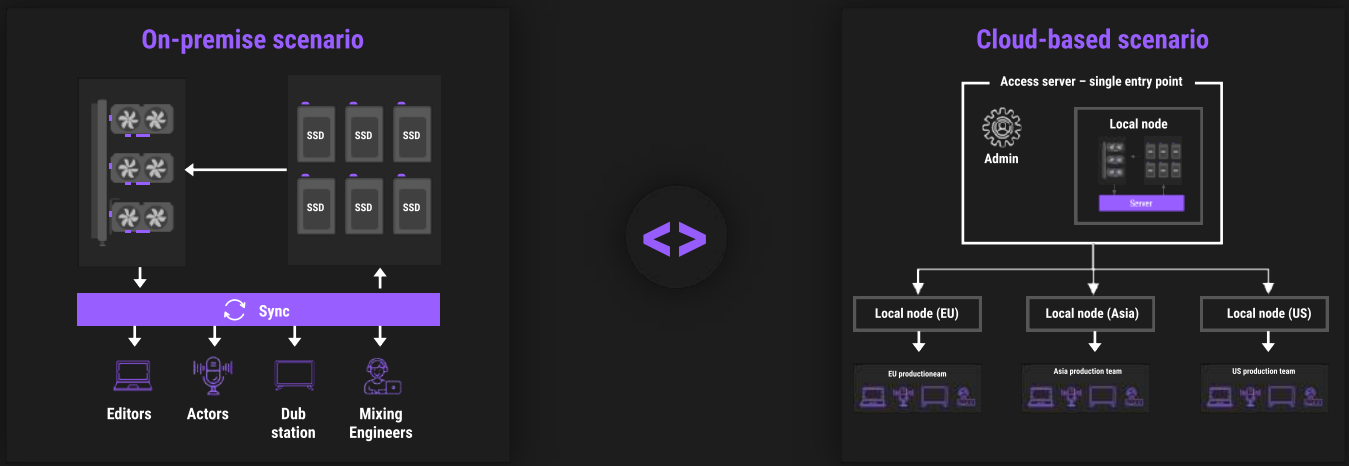


We can still rely on all our techniques for parallelization, especially as processors only work on a small number of input samples in parallel

We can now transfer even larger amounts of data at once and we do not have the 1ms delay requirement - so this is even the easier case

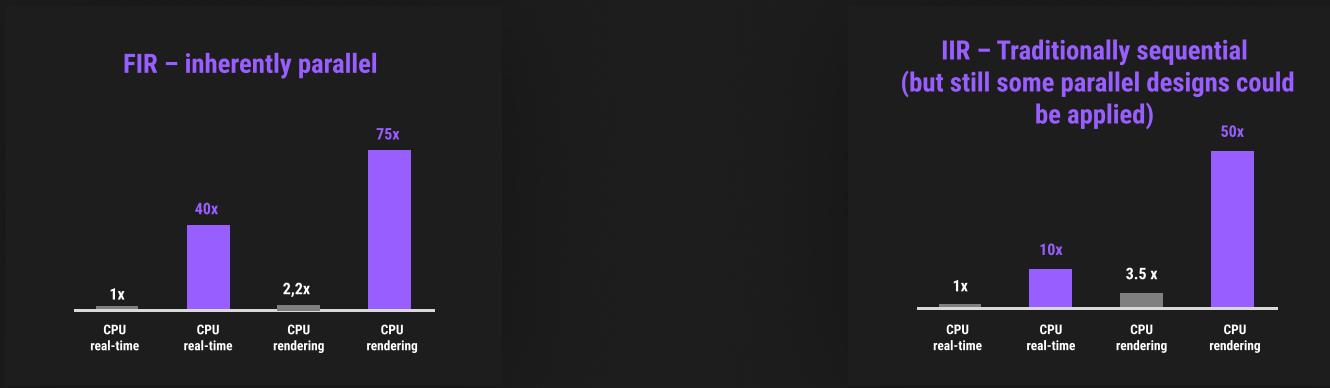
GPU-based solution can be easily scaled in all scenarios

On-Premise or cloud deployment solutions based on IT-first architecture



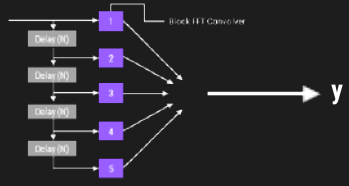
GPU Audio speedup over CPU implementations

Algorithms designed by GPU Audio achieve significant speedups over CPU processing in both real-time processing and rendering use cases, for both inherently parallel effects and traditionally sequential tasks



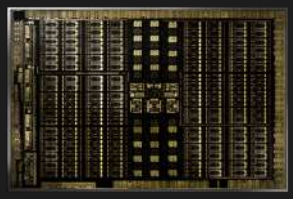
Example of easily parallelizable component: FIR

Definition $y[n] = b_0x[n] + b_1x[n-1] + b_2x[n-2] + ... + b_Nx[n-N]$



Finite impulse response filter or convolution due to the fact that all samples can be computed independently is easily parallelizable

However, the number of computations it costs even using non-trivial solutions is massive



Having so many cores in the modern GPU we can greatly beat CPUs in terms of executing lots of long FIR filters at the same time

Example of problematic component: IIR filter

Definition

Sequence: y[n] = b0 * x[n] + b1 * x[n - 1] + b2 * x[n - 2] - a1 * y[n - 1] - a2 * y[n - 2]

Transfer function: $H(z) = \frac{b_0 + b_1z^{-1} + b_2z^{-2}}{1 + a_1z^{-1} + a_2z^{-2}}$

Fundamental problem:

Output sequence y can be evaluated only sample by sample due to the fact that each output samples depends on previous two output samples (feedback) even if part or all input signal x is known at the start of the processing

Solution for sequential components like IIR filter

Using classical audio DSP designs we cannot use GPU efficiently, as, for example, our equalizer will be able to use only one thread per audio channel

Find design which is producing the same result has some degree of output dependency freedom

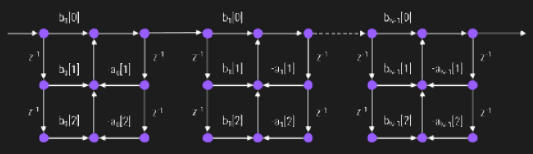
Requirements

- Retain original filter numerical stability and SNR
- Doesn't vastly increase number of computations that are needed to recalculate filter parameters
- Doesn't vastly increase number of computations that are needed to perform filtering
- Universally works for any IIR filter type

Further problem: cascade of IIR filters

Using classical audio DSP designs we cannot use GPU efficiently, as, for example, our equalizer will be able to use only one thread per audio channel

Many audio effects are formulated and implemented as a sequence of second order IIRs, for example equalizer is mostly the cascade of such filters, where the count of those filters depending on particular implementation can vary ~10-30 filters

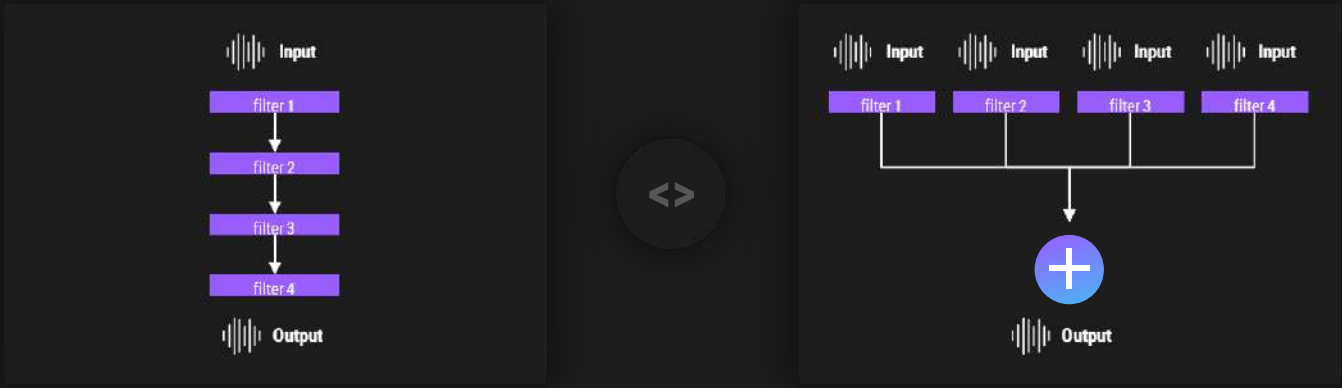


In the end we're getting sequence of filters and each filter internally also can compute output only sequentially

Solution for cascades

Develop topology of the audio processor to minimize sequential components,parallelize topology

This is not an easy task as we want to get the same audio output with completely different processor design



Engineering tricks for audio DSP parallelization

1

In compressor detection line we can change attack/release state only once 32 samples without any significant difference in the output

2

Our IIR filter implementation can compute two output samples at one iteration using 16 parallel and independent multiplications

3

Usage of custom derived equations in the form that greatly improves numerical stability in some critical places

Engineering tricks for audio DSP parallelization

Our equalizer has all of its IIR filters in parallel composition

Our IIR filter implementation can compute two output samples at one iteration using 16 parallel and independent multiplications

Our delay lines and circular buffers are designed in the way that access to every sample in it will cost exactly the same amount of instructions

GPU audio

October 2022

GPU Audio proprietary technologies

Meet the GPU Audio Rendering Engine powered by Scheduler

Our proprietary algorithm to solve these challenges

Scheduler Technology Overview

Host Scheduler

Multi-threaded lock-free low overhead design

- Chain Blueprint generation
- Communication with the DAW
- Data collection and combination
- Triggering processing on the GPU
- Execution time prediction

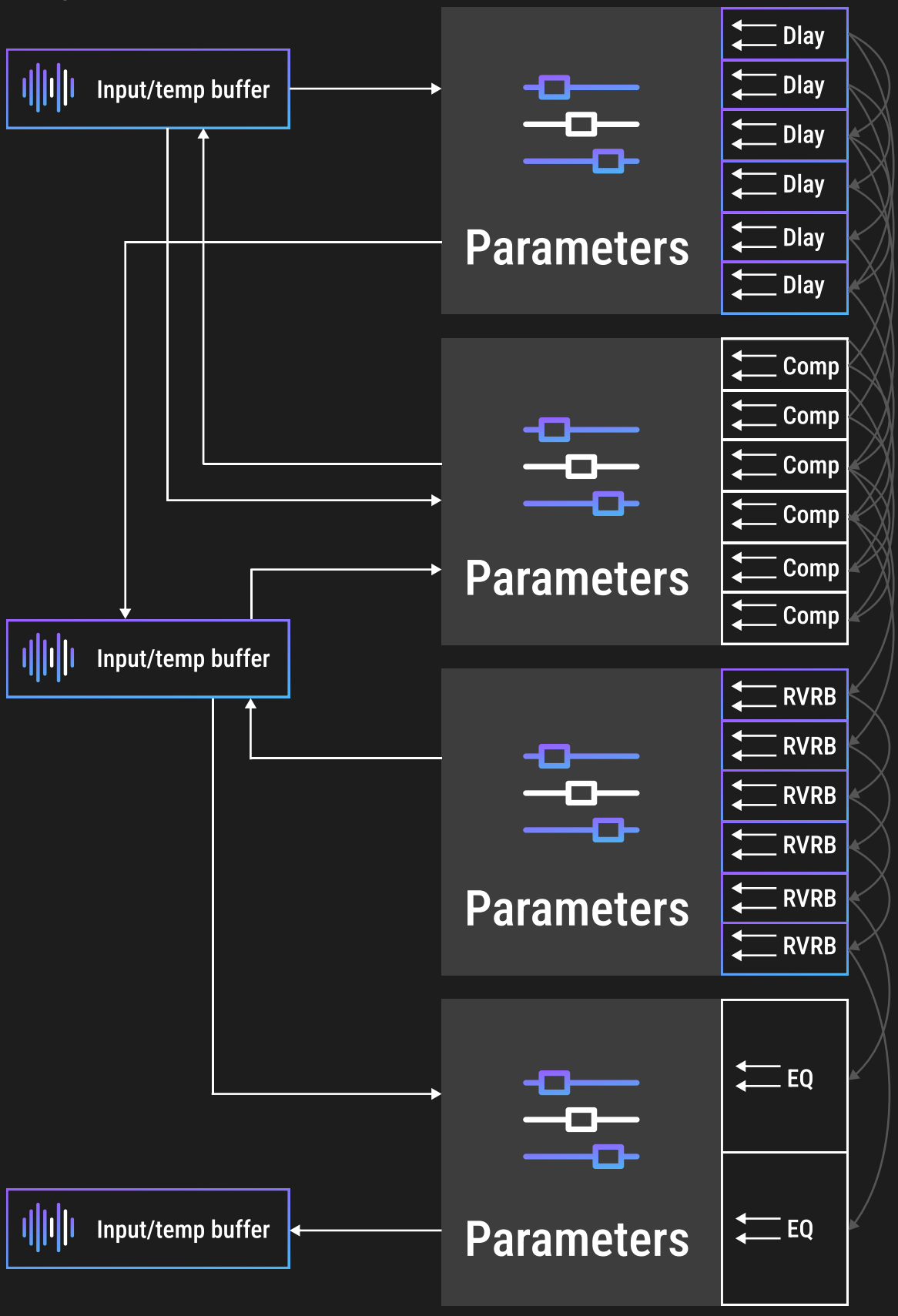
Device Scheduler

Dynamic, dependency-aware, priority-based, parallel, distributed scheduler, running across kernel launches

- Chain blueprint management
- Task collection
- Dynamic Resource management
- Dependency resolution
- Execution blocking

Scheduler Blueprints

Final blueprint



While the parameters and the buffer setup changes for every launch, the task descriptions remain the same every time. That is why we call it blueprints

Blueprint generation and instantiation

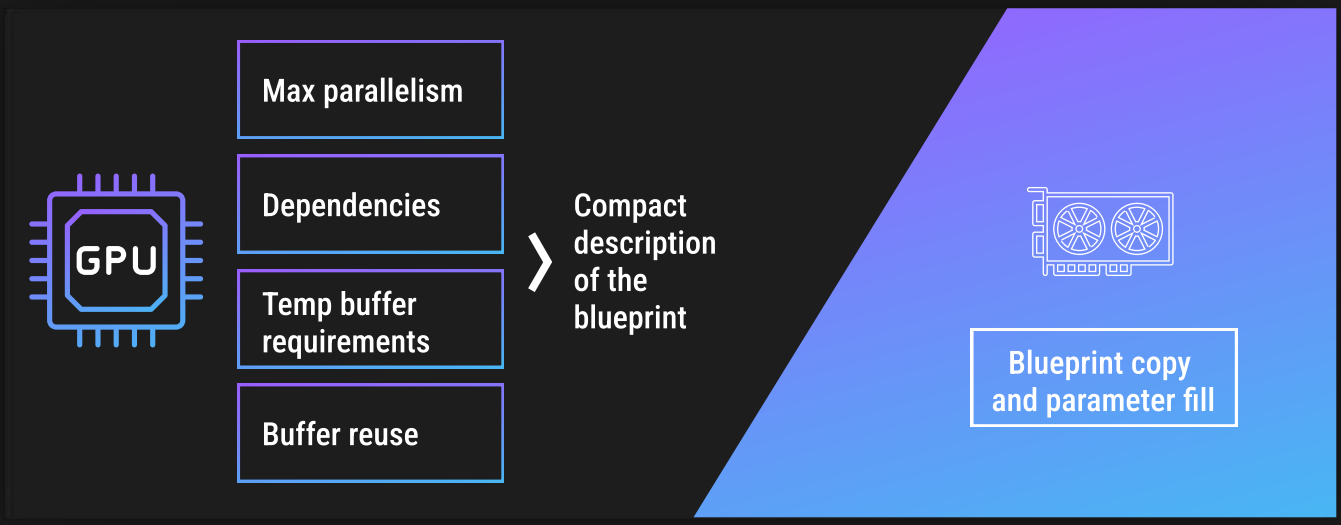
Blueprints are generated by the host scheduler and instantiated by the device scheduler

Host scheduler

- Derive parallelism from effects
- Determine parallelism across time and tasks
- Setup dependencies
- Determine temporary buffer requirements
- Find compact Blueprint description

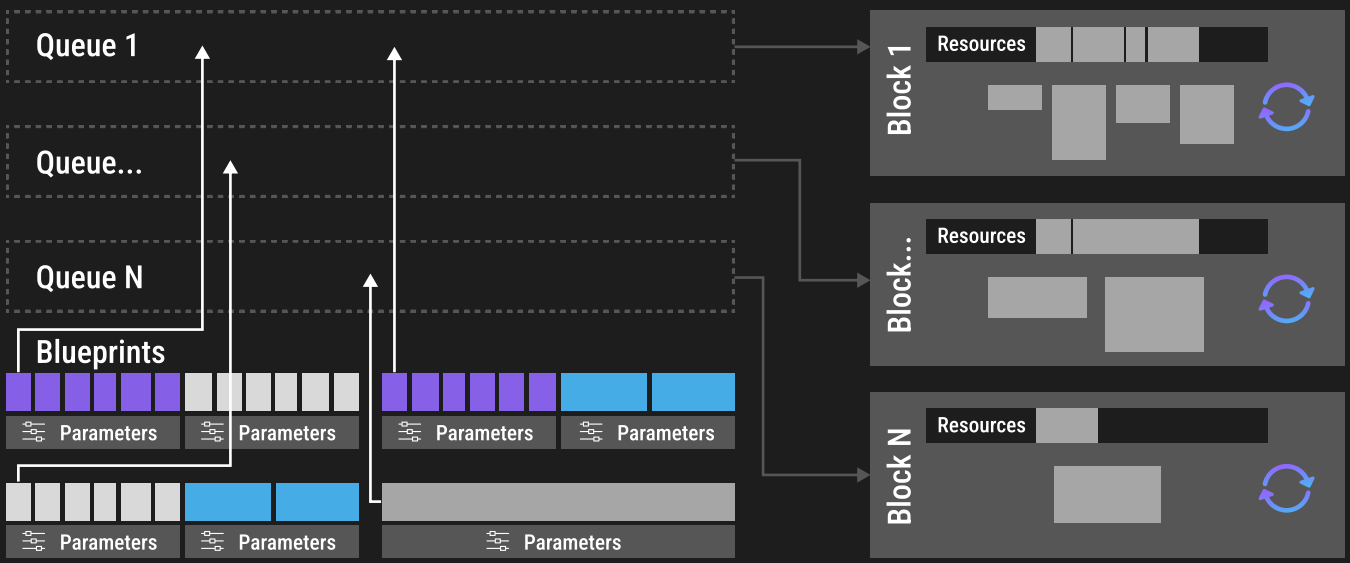
Device scheduler

- Receive input data
- Receive parameters
- Copy and instantiate Blueprint
- Fill Blueprint with new data
- Kick off processing
- Run task execution



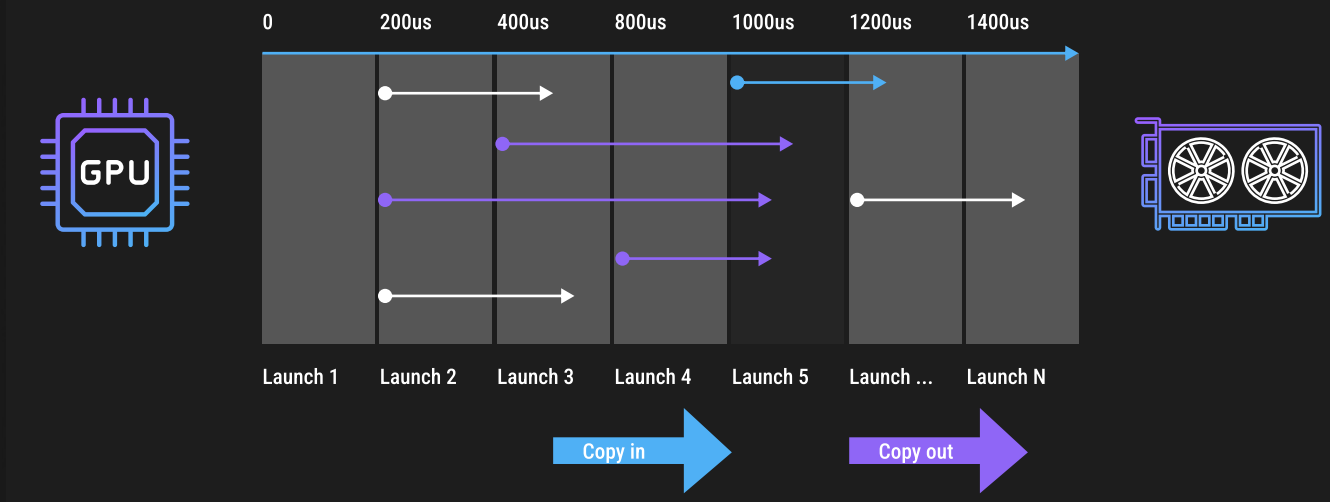
Scheduler running on Blueprints

Tasks from all blueprints are being processed in parallel

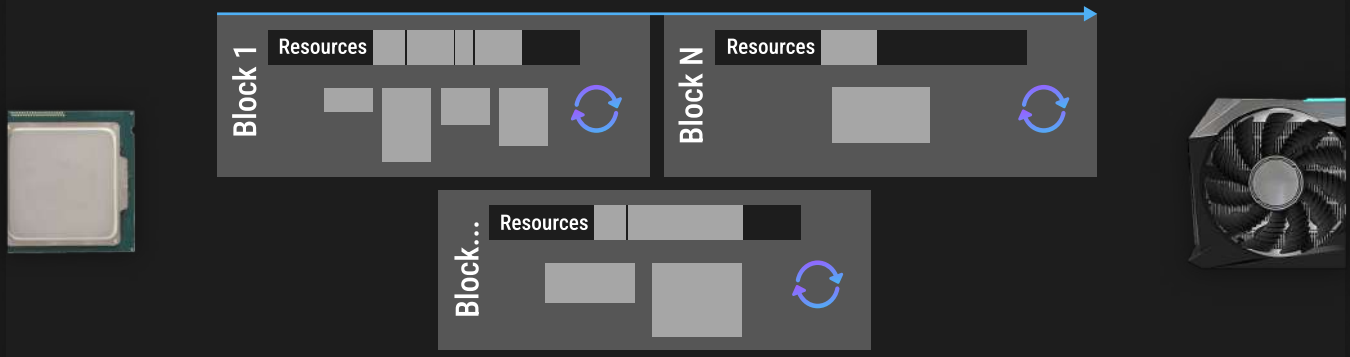


Pulsing Scheduler Design

1. Processing requests for different tracks are arising at any point in time
2. Discretize time into very short time windows and align processing with the windows
3. Memory transfers are merged for incoming and outgoing data



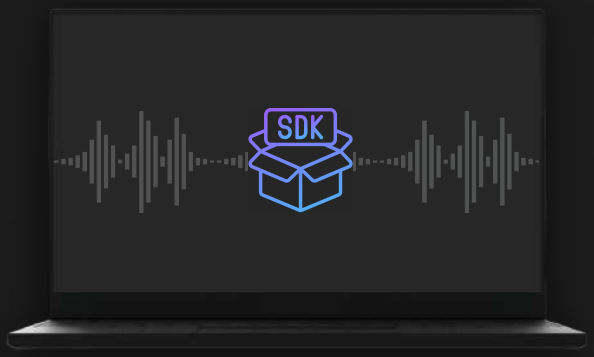
Device scheduler in each window



GPU Audio DSP SDK

Vast collection of unique algorithms which can be used to efficiently process sound on GPU

SDK for both enthusiasts and companies to build their own products using those algorithms



Well-known DSP processing kernels for easy implementation

Multi-layered – fits any use case

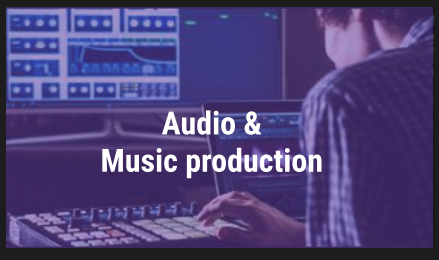
To summarize, the GPU Audio SDK is like an onion

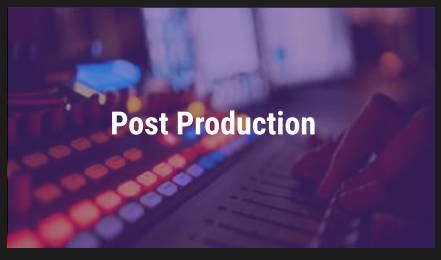
This sets developers up with incredible features, across any platform that uses audio

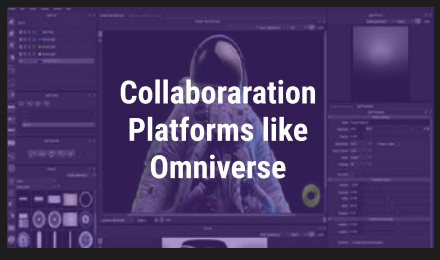
Many layers that can be used as desired

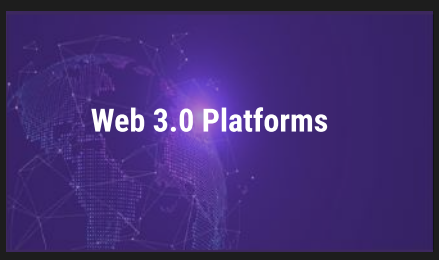
Low latency

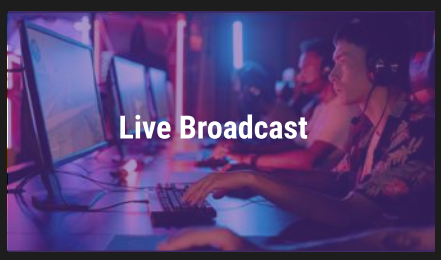
High performance DSP











October 2022

Roadmap

GPU Audio growing ecosystem

Early Access Launch

Feb 2022

Beta Suite Release

Mid 2022

MAC OS release

Nov 2022

Third Party Products (ex: Dave Pensado's "Magic Box" plugin, Mach1 Spatial System)

Q1 2023

More to come

Full Beta Suite Release
(7+ products on the roadmap)

Accelerate Cloudstage to Development: Our Flagship Platform for Post Production Enterprise

New DAWs

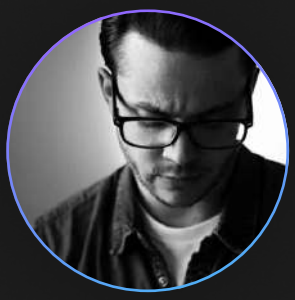
Monetization (After all beta products are out and developer community is established)

GPU Audio SDK

October 2022

Team

Team

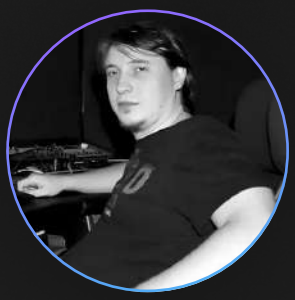


Jonathan Rowden

- Independent Sound Designer / Composer / Saxophonist
- Partnership Development

Alexander Talashov

- Technology Architect
- Managing Partner
- CS & HPC Scientist



Vasiliy Sumatokhin

- Managing Partner
- Sound Engineerfor almost 21 years

Markus Steinberger

- Senior Researcher, Advisor
- MSc in Computer Engineering,
- PhD in Computer Science

Aleksandrs Prokopcuks

- Managing Partner
- CTO
- DSP Scientist

GPU audio

Join us!

Wisit our website - www.gpu.audio

Learn more about our company on LinkedIn

Discord – Join the Community!